Validierung und Erweiterung von bestehenden „Floating Car Daten“ durch den „Traffic Message Channel“ Dienst des „Radio-Data-System“ zur Optimierung der Verkehrssituationsanalyse in Berlin.

Masterarbeit

von

David Suske

Technische Hochschule Wildau [FH]

Fachbereich Ingenieurwesen/Wirtschaftsingenieurwesen, Telematik

Eingereicht am: 28.10.2011

Erstbetreuerin: Frau Prof. Dr. rer. nat. Janett Mohnke

Zweitbetreuer: Herr Prof. Dr.-Ing. Anselm C. Fabig

Kurzfassung

Im Rahmen der Masterarbeit wird für das Land Berlin das bestehende FCD-Informationssystem des Deutschen Zentrums für Luft- und Raumfahrt (DLR) durch Informationen aus dem „Traffic Message Channel“ angereichert, um darauf aufbauend die Verkehrssituationsanalyse zu optimieren.

Hierzu wird ausgehend von den anfangs beschriebenen Grundlagen, für Floating Car Data (FCD) und Travel Message Channel (TMC) ein Verständnis für die unterschiedlichen Systeme vermittelt. Des Weiteren werden ableitend von der bestehenden Systemarchitektur des DLR, neue Dienste entwickelt, die eine verbesserte Übersicht über den aktuellen Verkehr in Berlin ermöglichen. Die neu erstellte TMC-Verarbeitungskette ist modular entworfen, so dass das erstellte System auf andere Städte oder Verkehrserfassungssysteme übertragen werden kann. Ferner wird gezeigt, wie TMC-Meldungen unter Berücksichtigung der gültigen ISO-Standards 14819-1 bis 14819-3 zu empfangen und zu interpretieren sind.

Darüber hinaus bietet diese Arbeit einen neuartigen Einblick in die TMC-Datenanalyse, der zu neuen Aussagen bezüglich der Qualität von TMC-Meldungen im urbanen Bereich Berlin führt. Ein weiterer, neuer Faktor ist die Darstellung der FCD und TMC-Daten, die erstmals in einer eigenen Webanwendung zusammengeführt dargestellt werden. So ist es nicht nur möglich, die neuen Inhalte weltweit online zugänglich zu gestalten, sondern auch einen völlig neuen Informationsgehalt, für die Verkehrssituationsanalyse zu generieren. Die neue Datenbasis ermöglicht erstmals eine auftretende Verkehrsstörung, mit einem Ereignis zu verbinden.

Ferner wird im Hinblick auf die Komplementierung und Anbindung an andere Dienste beschrieben, wie ein Webservice bereitgestellt werden kann, der den neuesten Techniken der Softwareentwicklung entspricht und die neuen Daten Nutzern in geeigneter Form zur Verfügung stellt.

Abstract

In the context of the master thesis, the existing FCD information system of the German Aerospace Center (GAC) for Berlin is enriched with information from the Traffic Message Channel, in order to optimize traffic condition analysis.

On the basis of the description for Floating Car Data (FCD) and the Travel Message Channel (TMC), which are described in the beginning of the thesis, an understanding for the different systems is presented. In the next step, deriving from the existing system architecture of the GAC, new services are developed, which improve the overview of current traffic situations in Berlin. The newly developed modular TMC process value chain can be easily transferred to other cities or traffic logging systems. Additionally, it is outlined how one should receive and interpret TMC messages, while considering the valid ISO Standards 14819-1 to 14819-3.

The thesis further offers a divergent perspective on the TMC data analysis, which leads to new acknowledgements concerning the quality of TMC messages in the urban area of Berlin. An emergent factor is the presentation of FCD and TMC data, which are, for the first time, commonly represented in a single web application. This way it is not only possible to offer new content world-wide, but to also increase accessibility while at the same time creating the ability to generate completely new information for the traffic condition analysis. The new basis of data, precedently, enables the connection of arising traffic obstructions with the corresponding event.

For the complementation with and connection to other systems, it is furthermore described how a web service can be provided, which corresponds to the newest software development techniques. This provides new data to stakeholders in a suitable form.

INHALTSVERZEICHNIS

Abbildungsverzeichnis

Tabellenverzeichnis

Diagrammverzeichnis

Abkürzungsverzeichnis

API:	Application Programming Interface
AJAX :	Asynchronous JavaScripting and XML
DLR:	Deutsches Zentrum für Luft- und Raumfahrt e.V.
ESRI:	Environmental Systems Research Institute
FCD:	Floating Car Data
FDP:	File Transfer Protocol
GNS:	Global Navigation Systems
GPRS:	General Packet Radio Service
GPS:	Global Positioning System
GSM:	Global System for Mobile Communications
HTTP:	Hypertext Transfer Protocol
JDBC-API:	Java Database Connectivity Application Programming Interface
JPEG:	Joint Photographic Experts Group (Norm ISO/IEC 10918-1)
JSON:	JavaScript Object Notation
LOS:	Level of Service
PNG:	Portable Network Graphics
RBDS:	Radio Broadcast Data System
RDS:	Radio Data System
REST:	Representational State Transfer
SOAP:	Simple Object Access Protocol
TDP:	Traffic Data Platform

Telnet:	Telecommunication Network
TMC:	Traffic Message Channel
UDDI:	Universal Description, Discovery and Integration
UML:	Unified Modeling Language
UMTS:	Universal Mobile Telecommunications System
UTC:	Coordinated Universal Time
WGS:	World Geodetic System
WKT:	Well-Known Text
W3C:	World Wide Web Consortium
XML:	Extensible Markup Language

Glossar

Alert-C

Alert-C ist ein europäischer Standard für sprachunabhängigen Austausch von Verkehrsinformationen via RDS-TMC Channel, die Referenz ist ISO 14819 [1].

American Standard Code for Information Interchange

Der American Standard Code for Information Interchange ist der Amerikanische Standard zur Codierung beim Datenaustausch [2]

Application Programming Interface

Die Programmierschnittstelle ist ein Sammlung von Funktionen und Methoden die es ermöglicht, eine Verbindung zu anderen Programmen herzustellen.

Approximieren

Approximation ist eine Näherung.

Asynchronous JavaScripting and XML

Das Asynchronous JavaScripting and XML beschreibt die Technologie für eine asynchrone Datenübertragung zur Darstellung von dynamischen Webinhalten. Durch diese Technologie ist es möglich nur Teile einer Webseite zuladen.

Austrosoft

Austrosoft ist eine Firma die Flottenmanagement-Systeme anbietet. Die Firma wurde 1982 von Michael Weiss und Gerhard Tillich gegründet. [3]

Axiom

Axiom ist ein Grundsatz oder auch fundamentales Prinzip, das als richtig und gültig angesehen wird.

Baudrate

Baudrate ist ein Maß für die Schrittgeschwindigkeit bei der Übertragung in der Nachrichtentechnik. Dabei entspricht 1 Baud der Geschwindigkeit, wenn 1 Zeichen pro Sekunde übertragen wird.

Coordinated Universal Time

Die Coordinated Universal Time beschreibt eine koordinierte Weltzeit, deren Grundformat mit Zeitversatz so aussehen kann: 20110525**T**142050+0100. Das Datum und Uhrzeit wären in

diesem Beispiel der 25.05.2011 und 13:20:50 Uhr. Demnach ist das Datum und die Zeit in folgender Reihenfolge zu interpretieren: Jahr, Monat, Tag, dann folgt ein großes „T“ für Time mit der Stunde, Minute, Sekunde und der Zeitversatz.

Deployment

Das Deployment beschreibt in der Informatik den Ablauf zur Installation einer Software.

Deutsches Zentrum für Luft- und Raumfahrt

Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR) in der Helmholz-Gemeinschaft Institut für Verkehrssystemtechnik am Standort Adlershof in Berlin, ist ein Institut für Verkehrsforschung.

Dummywert

Der Dummywert ist eine Größe, die nicht gemessen wurde (fiktiv), und lediglich als ein Platzhalter zu verstehen ist.

Environmental Systems Research Institute

Das Environmental Systems Research Institute ist ein US-amerikanischer Softwarehersteller. [4]

Extensible Markup Language

Die Extensible Markup Language ist ein durch das World Wide Web Consortium (W3C), in der XML-Spezifikation herausgegebener Standard.

File Transfer Protokoll

Das File Transfer Protokoll ist ein Datenübertragungsprotokoll im Internet.

Flag

Ein Flag entspricht in der Informatik einem Datentyp, der nur die Werte 0 oder 1 annehmen kann, also wahr (true = 1) oder falsch (false = 0).

Floating Car Data

Als Floating Car Data werden Daten bezeichnet, die aus einem Fahrzeug heraus generiert werden und Informationen zu Geschwindigkeit und Position enthalten.

General Packet Radio Service

Der General Packet Radio Service ist ein Dienst zur Datenübertragung in UMTS oder GSM.

Global Navigation Systems

Global Navigation Systems ist eine Firma, die basierend auf dem weltweit standardisierten Traffic Message Channel Hard- und Softwarelösungen anbietet.

Global Positioning System

Global Positioning System ist ein globales Satellitennavigationssystem. [5][S23]

Global System for Mobile Communications

Das Global System for Mobile Communications ist ein Mobilfunkstandard, der bei der Handykommunikation eingesetzt wird.

Hypertext Transfer Protocol

Das Hypertext Transfer Protocol ist eine Technik des Internets zur Datenübertragung, das vom W3C standardisiert wurde.

Java Database Connectivity

Java Database Connectivity ist eine Programmierschnittstelle für Java, die es dem Anwender ermöglicht, auf Datenbanken zuzugreifen.

JavaScript Objekt Notation

Die JavaScript Objekt Notation ist eine Textform wie XML, nur komprimierter. Dabei ist jedes JSON-Dokument in gültigem JavaScript geschrieben und kann einfach übersetzt werden.

JPEG

Die Joint Photographic Experts Group erstellt eine Bildkomprimierungsmethode, die in der Norm ISO/IEC 10918-1 als Standard definiert wird. [6]

Kausalität

Unter Kausalität (lat. Ursächlichkeit) versteht man den Zusammenhang zweier aufeinander folgender Ereignisse und dessen Abhängigkeit zueinander. [7]

Level Of Service

Der Level Of Service beschreibt Qualitätsstufen, die den Verkehrsfluss auf einer Straße beschreiben. Der Verkehr wird dabei in 6 Stufen unterteilt, wobei Stufe 1 oder auch A (je nach Literatur) eine sehr gute Bewegungsfreiheit der Verkehrsteilnehmer beschreibt und die Stufe 6 oder F eine Überbeanspruchung des jeweiligen Verkehrsverbindungsmediums definiert. [8]

Lichtsignalanlage

Eine Lichtsignalanlage oder auch Ampel dient der Steuerung des Straßenverkehrs durch verschiedenfarbige Lichter. [8][S45]

Link

Ein Link ist eine eingebettete Verknüpfung, die automatisch zu einem anderen oder dem gleichen Dokument an einer andern Stelle verweist.

Mapserver

Der Mapserver ist ein Server, der darauf spezialisiert ist, Services anzubieten, die Geodaten verarbeiten und bereitstellen können.

Navteq

Die börsennotierte Firma Navteq ist ein Anbieter von Geodaten, mit den daraus abgeleiteten Möglichkeiten für die Verarbeitung und Darstellung der Daten, wie einem Kartenmaterial. [9]

Portable Network Graphics

Portable Network Graphics ist ein Grafikformat für Rastergrafiken.

PostGIS

PostGIS ist eine Erweiterung für PostgreSQL, die die Möglichkeit, bietet geografische Objekte und Funktionen zu nutzen. [10]

PostgreSQL

PostgreSQL ist ein objektrelationales Open Source Datenbankmanagementsystem.

Radio Broadcast Data System

Das Radio Broadcast Data System ist die Bezeichnung für das Amerikanische Radio Data System (RDS).

Radio Data System

Das Radio Data System ist ein Datenübertragungsverfahren beim Hörfunk.

Representational State Transfer

Der Representational State Transfer ist ein Programmierstil für Webanwendungen, die keinem Standard unterliegen. Vielmehr werden die vorhandene Architektur und die Möglichkeiten des HTTP neu interpretiert.

Simple Objekt Access Protocol

Dieses Netzwerkprotokoll beschreibt einen Datenaustausch zwischen einzelnen Systemen in einem XML Format.

Status Quo

Der aus dem Latein entlehnte Begriff Status Quo steht für den aktuellen Zustand des im Kontext spezifizierten Mediums.

Struct

Ein Struct ist in der C++-Programmierung ein Schlüsselwort, um einen eigenen Datentyp anzulegen.

Telecommunication Network

Das Telecommunication Network ist ein vergleichbar altes Protokoll zum Datenaustausch im Internet, die Kommunikation ist vor allem Server-Client orientiert.

Thread

Ein Thread ist eine sequentielle Abfolge von Anweisungen in einem Programm, wichtig hierbei ist, dass mehrere Thread‘s parallel ablaufen können.

Traffic Data Plattform

Die Traffic Data Plattform ist ein Sammelbegriff für alle Daten und deren Datenbanken, die im Rahmen von FCD genutzt werden.

Traffic Message Channel

Der Traffic Message Channel ist ein Dienst des Radio Data Systems, um Verkehrsmeldungen an Empfangsgeräte zu senden.

Unified Modeling Language

Die Unified Modeling Language ist eine grafische Modellierungssprache, die genutzt wird, um komplexe System zu beschreiben. [11]

Universal Description, Discovery and Integration

Universal Description, Discovery and Integration, beschreibt einen standardisierten Verzeichnisdienst, der bei Service orientierten Diensten im Internet eingesetzt wird.

Universal Mobile Telecommunications System

Das Universal Mobile Telecommunications System ist ein Mobilfunkstandard der dritten Generation, auch als 3G abgekürzt, der bei der Handykommunikation eingesetzt wird.

Urbanisierung

Unter Urbanisierung (lat. urbs: Stadt) wird eine Zunahme des städtischen Raums verstanden, welche auch als Verstädterung bekannt ist. [8][S151]

Web Service Description Language

Die Web Service Description Language ist eine Protokoll unabhängige Beschreibungssprache für Netzwerkdienste.

Well-Known Text

Well-Known Text ist eine menschenlesbare Darstellung.

Wrapper

Ein Wrapper beschreibt eine Software die eine andere umhüllt. Der Zweck dieser Umhüllung ist es, die Softwarefunktionalität zu erhalten, wenn einem anderen System die gleiche Software zugänglich gemacht wird.

World Geodetic System

Das World Geodetic System (WGS-84) wurde 1984 als ein neues Geodaten-Referenzsystem mit einer einheitlichen Positionsangabe auf der Erde erdacht und beschreibt die Positionierung auf Basis eines Referenzellipsoiden.

World Wide Web

Das weltweite Netz ist ein Dienst des Internets, der es ermöglicht Hypertext Dokumente zu senden und zu empfangen.

World Wide Web Consortium

Das World Wide Web Consortium ist eine Gruppe die Standardisierungen für Techniken des World Wide Web vorantreibt.

1. Einleitung

Die Einleitung soll einen kurzen Einblick in die Problemstellung geben, dies wird in der Motivation herausgearbeitet. Ferner wird die Zielstellung bzw. der Zweck der Arbeit beschrieben. Abschließend folgt die Erklärung des Aufbaus allgemein und dessen Teilung in Theorie und Praxis.

1.1. Motivation

Die beginnende Urbanisierung[1] Anfang des 19. Jahrhunderts führte zu einer Erhöhung der Bevölkerungsdichte in den Städten. [12] Die zur gleichen Zeit einsetzende Industrialisierung erhöhte das Aufkommen von Automobilen auf den Straßen. Dies brachte unter anderem ein wachsendes Verkehrsproblem im Bereich der Verkehrsregelung mit sich. Dieses grundlegende Problem hat bis heute Bestand. [13] Zur Steuerung und Regelung des immer größer werdenden Verkehrsaufkommens benötigt man entsprechende Verfahren der Verkehrsdetektion. Das Erkennen einer aktuellen Verkehrssituation ist unabdingbar für eine effektive Regelung des Verkehrs.

Es kann kein Stau umfahren werden, der nicht bekannt ist!

Dieser scheinbar triviale Ansatz bringt die Kausalität[2] des Verkehrsgeschehens auf einen Nenner. Wenn man den Verkehr steuern will, so muss man ihn kennen. Ferner ist anzunehmen, dass eine verbesserte Detektion zu einer präziseren Steuerung führen würde. In unserer wirtschaftsorientierten Welt steht vor allem die Frage nach der Kosten-Nutzen-Analyse stark im Vordergrund. In diesem Zusammenhang erkannte Dr. Eckehard Schneider: „Radikal gesprochen, ist ein optimales Kosten-Nutzen-Verhältnis erst am Rande der kritischen Verkehrsdichte erreicht.“ [14]^{S355} Dabei bezieht er sich auf die Wirtschaftlichkeit solcher Detektoren, die, wie er konstatiert, nicht besonders gut sind. Doch der offensichtliche Nutzen bleibt bei einer rein kapitalistischen Betrachtung verborgen. Um dies besser verstehen zu können, soll ein kleiner Einblick in bestehende Verfahren gegeben werden. Diese Auswahl hat keinen Anspruch auf Vollständigkeit und dient lediglich dem allgemeinen Verständnis. [14]

[1] Unter Urbanisierung (lat. urbs: Stadt) wird eine Zunahme des städtischen Raums verstanden, welche auch als Verstädterung bekannt ist.
[2] Als Kausalität (lat. Ursächlichkeit) versteht man den Zusammenhang zweier aufeinander folgender Ereignisse und dessen Abhängigkeit zu einander. [7]

Induktionsschleifen

Das Verfahren bedient sich einer physikalischen Eigenschaft, der Induktivität(L). Diese ist die Fähigkeit einer Spule, ein Magnetfeld zu erzeugen, wenn eine Spannung angelegt wird. Bringt man entsprechende Spulen in die Fahrbahn ein, bewirkt jede metallene Masse, die in dieses Feld eintritt, eine Veränderung der Induktivität ΔL.

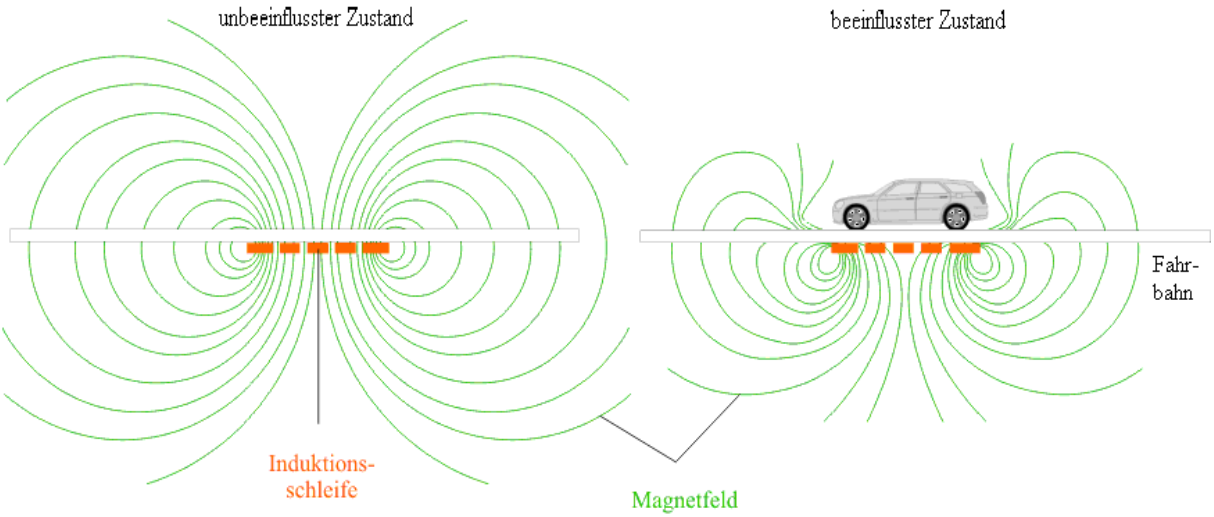


Abbildung 1: Induktionsschleifen im Straßenverkehr [15]

Die Abbildung 1 zeigt wie, ein PKW das Magnetfeld beeinflusst. Diese Veränderung der Induktivität ΔL ist messbar und kann in dem Verhältnis $\frac{\Delta L}{L} * 100$ ein Indikator für den jeweiligen Verkehrsteilnehmer sein. Bei einem PKW liegt dieses bei ca. 6 Prozent, bei Fahrrädern hingegen bei etwa 0,02 Prozent. Die gesammelten Daten dienen dabei nicht nur der Verkehrszählung. Eine weitere Anwendung ist die verkehrsabhängige Steuerung von Lichtsignalanlagen[3]. Bei einer Detektion eines Verkehrsteilnehmers über die Spule kann diese Information an die jeweilige Steuereinheit weitergegeben werden, um die Ampel auf Grün schalten, wenn es die Verkehrslage zulässt. [15]

Videosysteme

Ein an wichtigen Punkten angebrachtes Kamerasystem erstellt ein genaueres Bild der Geschehnisse vor Ort. Vor allem an Brücken, Tunneln und Hauptverkehrsadern wird diese Technik genutzt. Neben den Videobildern wird die mittlere Geschwindigkeit, Dichte des Verkehrs und die Anzahl der Fahrzeuge nach LKW o. PKW unterschieden. Geeignete

[3] Eine Lichtsignalanlage oder auch Ampel, dient der Steuerung des Straßenverkehrs durch verschiedenfarbige Lichter. [8][S45]

Methoden der Bildverarbeitung machen eine automatisierte Erfassung bei dieser Technik möglich, wie die folgende Abbildung 2 veranschaulicht. [16]

Abbildung 2: Verkehrskamera [16]

Die blauen Linien zeigen den detektierten Bereich. Wenn dieser in einer Ampelphase nicht frei wird, verändert sich die Farbe der Pfeile zu rot. Dies ist in der Abbildung 2 bei den Linksabbiegern zu sehen. Wenn die Rückstaulänge der Linksabbieger in der nächsten Ampelphase wieder zu groß ist bzw. nicht abgebaut werden kann, dann wird der rote Pfeil etwas länger. Somit ist es möglich, an den Farben der Pfeile die Auslastung der Fahrspur zu erkennen. Des Weiteren kann an dessen Länge die Zeit der Störung abgeschätzt werden. Die genauen Daten werden in einer Datenbank gespeichert, das Bild dient lediglich der Grobeinschätzung der Verkehrssituation. [16]

Floating Car Data

Dieses Verfahren setzt auf einen approximativen[4] Ansatz. Die Näherung für die gesamte Verkehrslage eines urbanen Bereiches wird auf der Basis eines einzelnen Verkehrsteilnehmers erstellt. Dieser einzelne Teilnehmer sendet seine Positions- und Geschwindigkeitsdaten an einen zentralen Punkt. Abgeleitet von dessen Geschwindigkeit kann auf die Gesamtgeschwindigkeit aller Verkehrsteilnehmer in diesem Bereich geschlossen werden. So geht diese Grundannahme davon aus, dass, wenn ein PKW auf einem gewissen Streckenabschnitt steht, auch andere an dieser Stelle stehen müssen. Warum dies nicht immer der Fall sein muss, soll im Kapitel 7. Abschnitt 1. näher erklärt werden. Ferner gilt: Je mehr einzelne Teilnehmer ihre Daten zu Verfügung stellen, desto größer und genauer wird der detektierte Bereich und deren Abschätzung. Da dieses Verfahren im weiteren Verlauf der Arbeit noch detailliert beschrieben wird, soll diese kurze Beschreibung vorerst genügen.

[4] Approximation ist eine Näherung.

DynVE (Dynamische und mobile Verkehrserfassung)

DynVE bezeichnet ein Verfahren zur mobilen, dynamischen und anonymen Verkehrserfassung und zählt wie FCD zur Kategorie der mobilen Erfassungsmethoden. Es setzt auf das Prinzip der Fremdortung und stellt somit einen gegenteiligen Ansatz zum Floating-Car Prinzip dar. Das Verfahren befindet sich im DLR-TS[5] im Entwicklungsstadium. Aus diesem Grund folgt nur eine kurze Erklärung zu Abbildung 3.

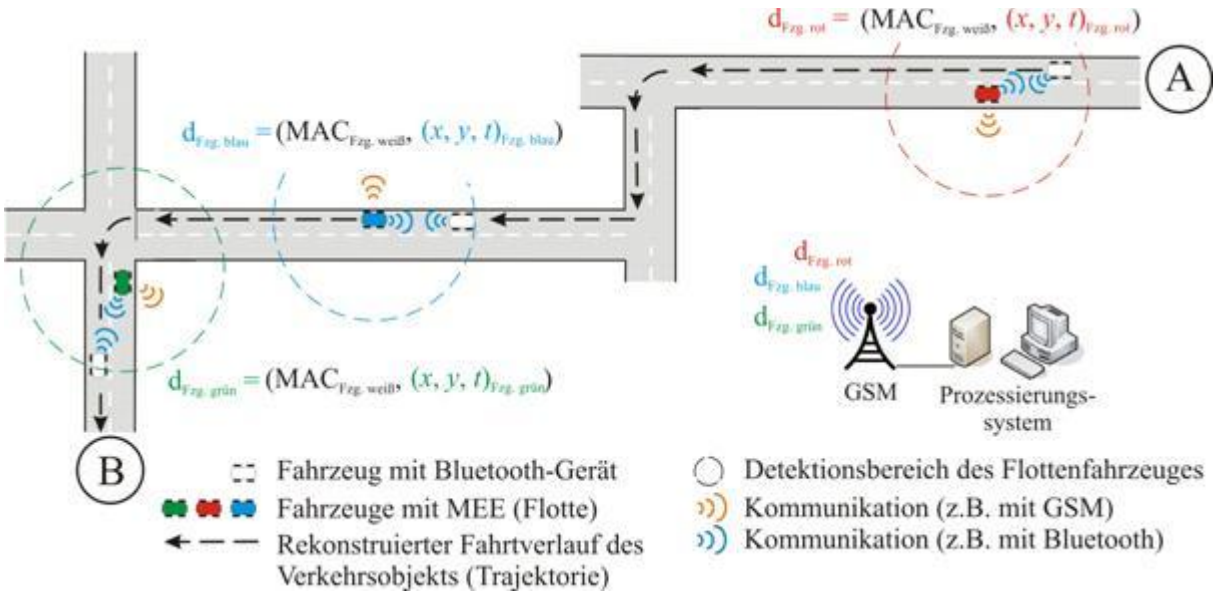


Abbildung 3: DynaMiC [17]

Das weiße Fahrzeug ist mit einer Erfassungseinheit ausgestattet und bewegt sich von Punkt A nach Punkt B. Somit ist es in der Lage andere Verkehrsteilnehmer zu detektieren. Der für die jeweiligen Autos dargestellte Erkennungsradius soll den Bereich kennzeichnen, in dem ein möglicher Teilnehmer erkennbar ist. Der Vorteil gegenüber der Selbstortung ist, dass wesentlich mehr Verkehrsteilnehmer anonym detektiert werden können. Die gesammelten Verkehrsinformationen können vom Detektionsfahrzeug z.B. über GSM[6] zum Prozessierungssystem geschickt werden. [17]

Die hier vorgestellten Systeme der Verkehrserfassung generieren eine Datenbasis, mit welcher der Verkehr erfasst und in Folge dessen gesteuert werden kann. Die Ausgangssituation für diese Betrachtung war die Frage nach der wirtschaftlichen Ineffizienz

[5] Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR) in der Helmholz-Gemeinschaft Institut für Verkehrssystemtechnik.

[6] Global System for Mobile Communications (GSM) ist ein Kommunikationsstandard, der vor allem bei der Handykommunikation eingesetzt wird.

solcher Systeme. Wie Dr. Eckehard Schneider bemerkt, ist eine Anlage für sich unwirtschaftlich, bzw. ist sie nur wirtschaftlich, wenn diese sehr gut ausgelastet wird. [14][S354] Diese Erkenntnis bestärkt das Bestreben, mögliche Detektoranlagen an stark befahrenen Punkten zu errichten. Die hierfür verwendeten Systeme sind die beiden erst genannten, Induktionsschleifen und Videosysteme. Dem Mangel dieser Techniken an Wirtschaftlichkeit kann entgegengewirkt werden, indem die Entwicklung von dynamischen Verfahren vorangetrieben wird. Die beiden zuletzt genannten Verfahren, Floating Car Data und DynVE, sind solche dynamischen Ansätze, wobei DynVE noch nicht im Regelbetrieb beim DLR integriert ist. Zur Verbesserung der Ausgangsbasis beim DLR, für eine genauere Verkehrslagedarstellung und dessen Analyse, wird überlegt, wie mehr Verkehrsinformationen zu möglichst geringen Kosten erhoben werden können. Diese Überlegung führt zu dem Traffic Message Channel, im Folgendem abgekürzt mit TMC. Der Dienst wird über das Radio Data System kurz RDS bereitgestellt, dazu mehr im Kapitel 3. „Grundlagen“. [14]

Überdies ist eine Erfassung einer flächendeckenden Verkehrslage in Großstädten und Ballungsräumen mit konventioneller Technologie, wie zum Beispiel lokalen Messschleifen und Infrarotsensoren, nur stichprobenartig möglich. Darüber hinaus erlaubt eine lokale Verkehrslageerfassung, insbesondere in städtischen Gebieten, nicht die Messung von Reisezeiten und Reisegeschwindigkeiten, welche weitere ergänzende und wertvolle Messgrößen darstellen. Einen Ausweg aus dieser Situation bieten die im Verkehr mitfließenden Fahrzeuge, die die schon benannten Floating Car Data (FCD) liefern können. Seit 2001 wird im DLR an der Nutzung dieser Technologie geforscht und entwickelt, so dass inzwischen ein ausgereiftes System zur Verfügung steht, in dem dennoch ein erhebliches Potential zur Weiterentwicklung steckt. [18]

Das DLR erzeugt aus über 4000 Berliner Taxis eine flächendeckende Verkehrslage für den urbanen Bereich. Diese flächendeckenden Informationen sind nicht in den bisherigen TMC-Daten enthalten und auch nicht in dieser Granularität durch TMC abbildbar.

Die Erweiterung der bestehenden FC-Daten durch TMC-Meldungen ist dabei nicht nur eine kostengünstige Ergänzung der bestehenden Datenbasis. Die Anreicherung führt ebenso zu einer völlig neuen Möglichkeit der Analyse des Verkehrsgeschehens, dass erstmals mit konkreten Ereignissen verknüpft werden kann. Somit ist es nicht nur möglich eine Verkehrsstörung, wie einen Stau zu erkennen, sondern auch dessen Grund in die Verkehrsanalyse einfließen zu lassen.

1.2. Zielstellung

Validierung und Erweiterung von bestehenden „Floating Car Daten“ durch den „Traffic Message Channel“ Dienst des „Radio-Data-System“ zur Optimierung der Verkehrssituationsanalyse in Berlin.

Wie aus dem Thema ersichtlich, soll im Rahmen der Masterarbeit für Berlin das bestehende FCD-Informationssystem des Deutschen Zentrums für Luft- und Raumfahrt (DLR) durch Informationen aus dem Traffic Message Channel angereichert werden, um die Verkehrssituationsanalyse zu optimieren. Der Schwerpunkt liegt dabei auf der Erstellung einer neuen Verarbeitungskette, die eine Erweiterung der bestehenden Datenbasis ermöglicht.

Zum einen sollen Meldungen über angekündigte Störungen, wie Baustellen und Sperrungen, von bestimmten Straßenabschnitten, die schon im Voraus mittelfristig bekannt sind (z.B. bei Großveranstaltungen), als TMC-Nachrichten berücksichtigt werden. Zum anderen sollen auch kurzfristig eintreffende Unfallmeldungen, Sperrungen und Stauwarnungen (teilweise inklusive Angabe der Reisegeschwindigkeit) mit den aus dem FCD-System generierten Werten verglichen und zu einer insgesamt verbesserten Verkehrsinformation zusammengeführt werden. Der Mehrwert dieser Konsolidierung ist eine Validierung und Erweiterung der bestehenden FCD-Daten. So ist es beispielsweise möglich, dass eine TMC-Meldung empfangen wird, die einen Stau meldet, welcher im FCD-System nicht bekannt ist. Wiederum besteht die Möglichkeit, dass ein von TMC gemeldeter Stau im FCD-System nicht bekannt ist. Ferner ist die Frage zu beantworten:

Was ist wenn FCD und TMC gegensätzliche Verkehrsmeldungen enthalten?

Die Beantwortung dieser Frage soll in einer Heuristik beschrieben werden, die als Umsetzung direkt in einem Prozessablaufplan die Möglichkeiten der Verarbeitung beschreibt.

Die Synergie der Meldungen führt zu einem verbesserten Bild der Verkehrslage, welches wiederum zu einer verbesserten Regelung führt. Des Weiteren soll die neue Verarbeitungskette modular entworfen und auf andere Systeme übertragbar gestaltet werden, um eine flexible und skalierbare Nutzung zu ermöglichen.

1.3. Aufbau und Vorgehensweise

Diese Arbeit teilt sich in einen theoretischen und einen praktischen Teil, die im Folgenden kurz vorgestellt werden.

Theoretischer Anteil der Arbeit:

Der theoretische Anteil bezieht sich vor allem auf die Erarbeitung der Grundlagen, um eine neue Datenbasis zu erstellen, so wird im Kapitel 3. „Grundlagen“ die beiden Datenarten genau charakterisiert. Ferner wird gezeigt, wie diese erfasst und auf welchem Wege sie zur Verfügung gestellt werden. Die Anforderungsanalyse Kapitel 4. umfasst den momentanen Status der FC-Daten, von der Generierung über die Verarbeitung bis hin zur Darstellung der im System befindlichen Daten. Im Folgenden ist der Soll-Zustand zu definieren. Was soll erreicht werden und welche Mittel bieten sich aus welchen Gründen an. Des Weiteren werden Probleme definiert, die während der Arbeit auftreten können, siehe Abschnitt 4.4. „Problemanalyse“. Der Vergleich der TMC-Meldungen mit den FC-Daten durch eine strukturierte Analyse und Erarbeitung von Konzepten soll eine neue verbesserte Datenbasis für das Verkehrsinformationssystem schaffen, siehe Kapitel 7. „Generierung einer neuen Datenbasis“.

Praktischer Anteil der Arbeit:

Dieser Teil zeigt die Umsetzung der zuvor formulierten Aufgaben. Hierfür soll erst der TMC Datenstrom aus dem RDS-Signal im GNS[7] 3.0 Standard decodiert werden. Der GNS 3.0 Standard ist mehr als ein De-facto-Standard zu sehen, da in vielen TMC-Empfängern ein GNS-Hardwaremodule verbaut wird [19], siehe Kapitel 5. „Empfangen der TMC-Meldungen“. Ferner gibt es ein Modul zur Prozessierung, welches im Abschnitt 5.2. „TMC Process Unit“ näher erklärt wird. Anschließend werden die Ergebnisse in geeigneter Form präsentiert und online als neuer TMC-Datenstrom bereitgestellt, dies soll in Kapitel 8. „Darstellung der neuen Datenbasis“ und Kapitel 9. „Webservice für neue Datenbasis“ beschrieben werden. Nach der Erstellung der Einzelkomponenten sollen diese in ihrem Zusammenspiel in einer kurzen Testphase überprüft werden, dies wird in Kapitel 10. „Testphase“ beschrieben.

[7] Global Navigation Systems (GNS) ist eine Firma, die basierend auf dem weltweit standardisierten Traffic Message Channel Hard- und Softwarelösungen anbietet.

Die Abarbeitung dieser beiden Anteile soll sich an der zu erstellenden Verarbeitungskette orientieren und der rote Faden durch diese Arbeit sein. Hierzu werden erst die Grundlagen für das Verstehen gegeben und anschließend sollen folgende Punkte in der beschrieben Reihenfolge abgearbeitet werden: Empfangen und Erstellen der TMC-Nachricht, Entwicklung eines Softwaremoduls zur Generierung einer neuen Datenbasis. Anschließend folgt die Auswertung der empfangenen Daten mit darauf folgender Visualisierung auf dem Web Frontend. Abschließend wird die Verarbeitungskette, die bereits erwähnte Testphase bzw. Inbetriebnahme der neuen Prozessierungskette einen zusammenführenden Überblick verschaffen.

2. Formulierung der Aufgabe und deren Restriktion

Die Anreicherung der bestehenden FCD-Informationen durch TMC-Daten unterliegt einigen Einschränkungen, die im Folgendem erwähnt werden sollen, um spätere Fragen auszuschließen.

Fehlerhäufigkeit

Die Richtigkeit der TMC-Meldungen ist von deren Herausgeber abhängig, den Radiosendern. Somit ist anzunehmen, dass einzelne Meldungen falsch oder veraltet sind. Die vorrangegangen Recherchen zu diesem Thema liefern keine genauen Zahlen der Fehlerrate für TMC-Meldungen. Dies verwundert nicht, da sich solch eine Aussage auf den Radiosender selbst beziehen müsste. Im Kapitel 6. „TMC Datenanalyse“ soll der Versuch unternommen werden, die Qualität der einzelnen Radiosender zu vergleichen und eine Abschätzung der Fehlerrate für einen Sender bestimmt werden.

Steuerungsorgane

Es gibt keine Zentrale, die die Einheitlichkeit der TMC-Meldungen für den urbanen Raum Berlin überprüft bzw. garantiert, dies wird zu Unterschieden in der Qualität und Quantität hinsichtlich der Meldungen zwischen den Radiosendern führen. Die Konsequenz ist, dass die Auswahl des richtigen Senders entscheidend zur Erweiterung der Daten beitragen wird, dies beeinträchtigt wiederum aufbauende Verfahren.

Aufnahmegeräte

Des Weiteren ist nur ein TMC-Empfänger bestellt worden. Dies führt dazu, dass die TMC-Meldungen von nur einem Radiosender zu einem genauen Zeitpunkt empfangen werden können. So ist es wiederum unmöglich, alle TMC-Meldungen des urbanen Bereiches Berlin zu einem Zeitpunkt zu erfassen und in die Verarbeitung miteinfließen zu lassen.

Zusammenfassend soll aus den vorliegenden Fakten folgender Sachverhalt definiert werden. Im Rahmen dieser Arbeit wird es nicht möglich, sein alle TMC-Meldungen für den urbanen Bereich Berlin zu erfassen. Ferner werden Aussagen von TMC-Meldungen zum aktuellen Verkehrsgeschehen eine, wenn auch durch diese Arbeit eingeschränkte (siehe Kapitel 6.), undefinierte Fehlerrate besitzen. Darüber hinaus werden die erhobenen Daten in dem zur Verfügung stehenden Zeitraum von maximal 6 Monaten, abzüglich der Entwicklungszeit von etwa 3 Monaten, nicht ausreichen, um eine exakte Analyse der einzelnen Radiosender bzw. deren TMC-Meldungen vorzunehmen. Vielmehr ist eine Abschätzung der Fehlerrate für die Radiosender zu erwarten.

Des Weiteren wird in dieser Arbeit beschrieben, wie eine neue Übersicht der Verkehrslage bzw. eine neue Datenbasis erstellt werden kann. In dieser Arbeit wird nicht betrachtet, wie dessen neue Informationen zu analysieren sind bzw. wie eine Verkehrssituationsanalyse auf Grundlage dieser neuen Basis aussehen kann!

Diese Aussage ist sehr wichtig, da beim Verstehen des Themas der Arbeit, dies vielleicht erwartet werden könnte.

Validierung und Erweiterung von bestehenden „Floating Car Daten“ durch den „Traffic Message Channel“ Dienst des „Radio-Data-System“ zur Optimierung der Verkehrssituationsanalyse in Berlin.

Der Schwerpunkt der Arbeit liegt auf der Validierung und Erweiterung von FCD durch TMC, also der Erstellung einer neuen Verarbeitungskette, die eine neue Datenbasis generiert. Zusätzlich soll in dem Kapitel 11. „Resultate“ betrachtet werden, wie und in welchem Maße die neuen Meldungen bzw. die neue Verkehrssituationsübersicht zur Optimierung der Verkehrssituationsanalyse beiträgt. Wichtig ist es, dass nicht auf die Analyseverfahren selbst eingegangen werden soll, da dies in der Aufgabenstellung nicht definiert wurde, siehe Kapitel 1. Abschnitt 2.

3. Grundlagen

In diesem Kapitel werden Grundlagen über die verwendeten Techniken, Dienste und Verfahren vermittelt. Darüber hinaus wird ein solider Fundus an Wissen erarbeitet, der als Basis für weitere Kapitel dient. Des Weiteren werden wichtige Begriff erklärt und in einen sinnvollen Kontext gebracht.

3.1. Radio Data System

Das Radio Data System, im Folgenden RDS abgekürzt, ist die Basis bzw. das Überträgermedium des Traffic Message Channel (TMC) und unabdingbar für ein umfassendes Verständnis. Vereinfacht beschrieben ermöglicht dieses System die Übertragung von Zusatzinformationen im Hörfunk. [20]

3.1.1. Einführung

Als am 8. November 1898 Tesla sein Roboterboot patentieren ließ, legte er den Grundstein der heutigen Radiotechnik. [21] So wurde 1920 der erste regelmäßige Radiobetrieb in Königs Wusterhausen begonnen. Diese Grundlagen führten zu dem Konzept des RDS welches 1983 von der Europäischen Rundfunkunion (EBU) entwickelt wurde. Ein Jahr später startete der Versuchsbetrieb, der 1987 in der Marktreife gipfelte. Im folgenden Jahr am 1. April 1988 wurde das System endgültig und offiziell eingeführt und in der Norm DIN EN 62106 standardisiert. [20] Die aktuell gültige Version ist die EN 62106:2010-07 (dt. Version), welche die Version von 2002 ersetzt und im Folgenden als Quelle dienen soll. [20]

3.1.2. Technische Grundlagen

Die Datenrate von RDS beträgt 1,1875 Kilobits pro Sekunde. Zur Modulation wird QAM (Quadraturamplitudenmodulation) eingesetzt. Dieses Verfahren vereint die Amplitudenmodulation mit der Phasenmodulation. Die Idee hinter dieser Kombination ist es, zwei Signale jeweils per Amplitudenmodulation auf einen Träger gleicher Frequenz zu bringen, welche um 90° verschoben sind. Die in Phase gebrachten Signale können somit unabhängig voneinander empfangen werden. [22]

3.1.3. Aufbau der RDS Nachricht

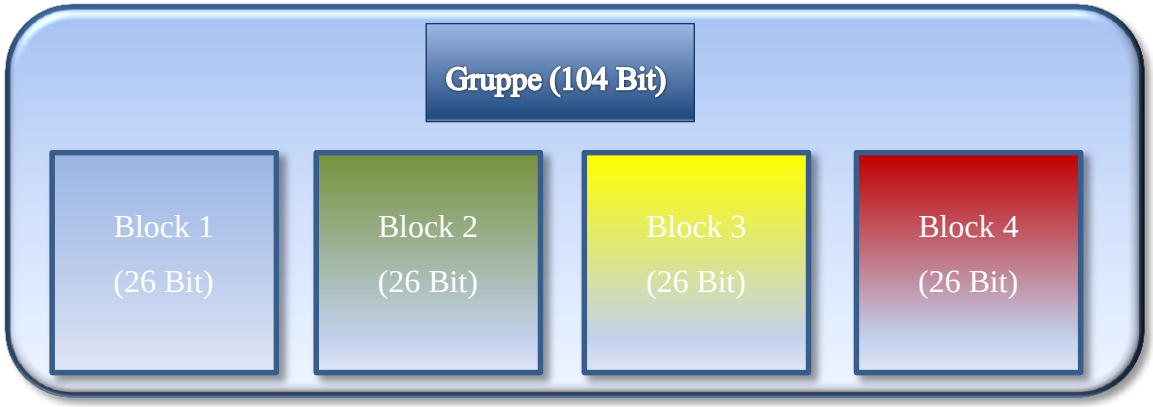


Abbildung 4: RDS Gruppe

Die kleinste RDS Einheit ist eine Gruppe, sie besteht aus 104 Bit zu je 4 Blöcken. Jeder Block hat 26 Bit, wie die Abbildung 4 zeigt.

Die Blöcke werden in den eigentlichen Informationsteil und dessen Prüfung aufgeteilt. Die Abbildung 5 „RDS Block“ stellt dieses Verhältnis dar. [20]

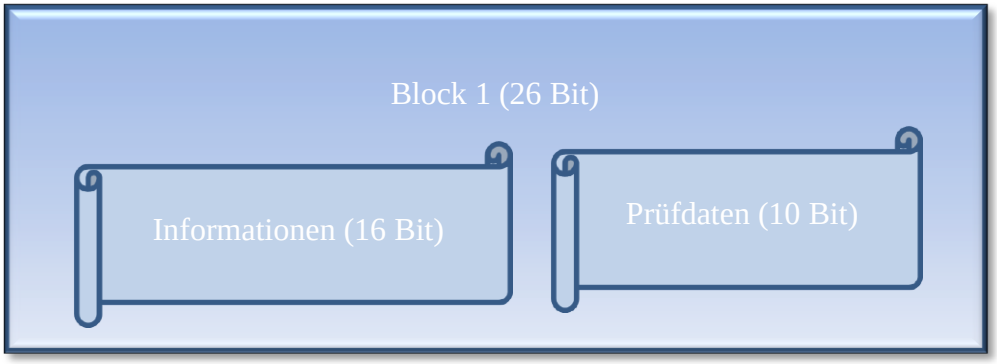


Abbildung 5: RDS Block

Wie in der Abbildung 5 zu sehen ist, tragen 16 Bit die eigentliche Information und 10 Bit werden als Fehlererkennung genutzt. Der gesamte Informationsgehalt einer Gruppe ist demnach 4-mal 16 Bit, also 64 Bit Informationen. Um die einzelnen Informationsbits unterscheiden zu können, werden ihnen Variablen zugeordnet. [20]

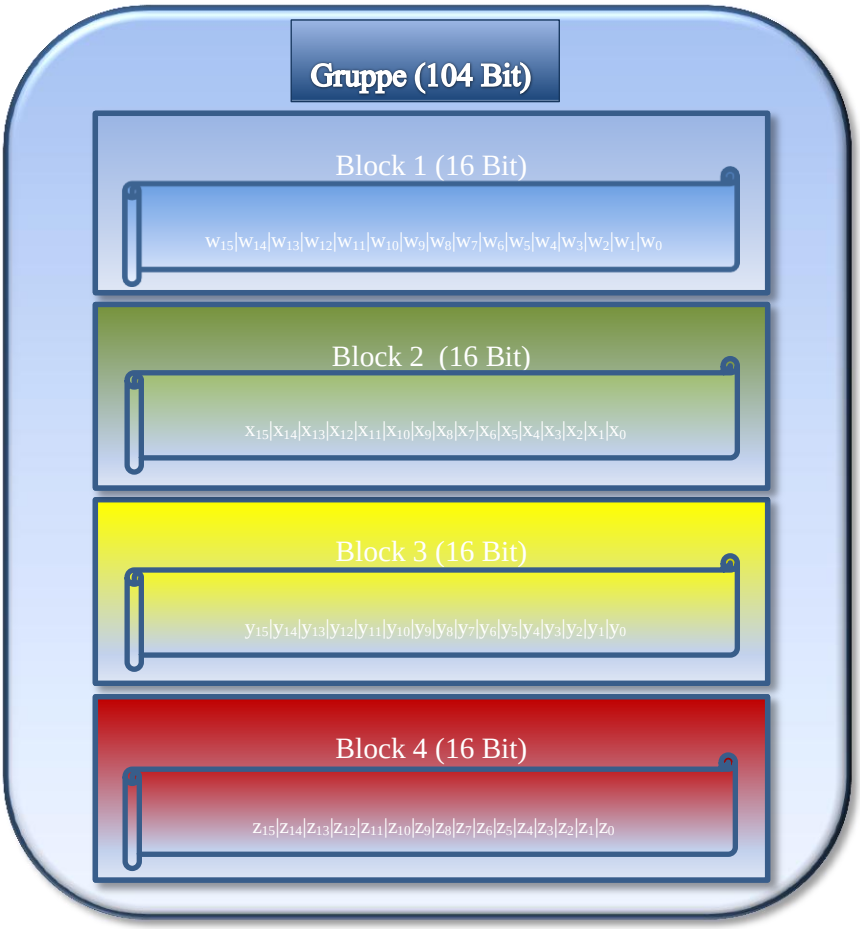


Abbildung 6: RDS Gruppe Variablenidentifikation

Die Abbildung 6 „RDS Gruppe Variablenidentifikation“ zeigt, wie die einzelnen Bits eindeutig gekennzeichnet werden können. Für die weitere Interpretation der Information werden die 16 Bit in je 4 Bits in ein Hexadezimalformat gebracht, wie die Abbildung 7 für den ersten Block zeigt.

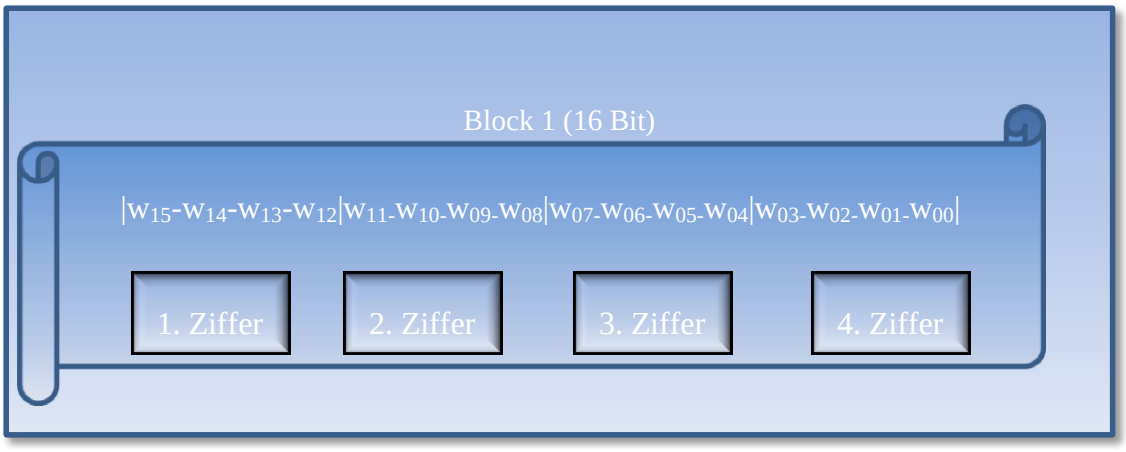


Abbildung 7: RDS Block in Ziffern

Der hier beschriebene erste Block der RDS Gruppe enthält dabei den Programm-Identifikationscode (PI-Code). Dieser ist in jeder Nachricht enthalten und im ersten Block zu

finden. Ferner hat jede der Ziffern ihre Bedeutung, der erste Hexadezimalwert bezeichnet das Land, der zweite die Größe des Sendebereiches, Nummer 3 das Bundesland oder Zähler und der 4. Wert identifiziert den Sender. Wenn der entsprechende RDS Block von einem landesweiten Sender ausgestrahlt wird, dann fungiert Wert 3 als der erwähnte Zähler. Ein einfaches Beispiel soll dies verdeutlichen. [20]

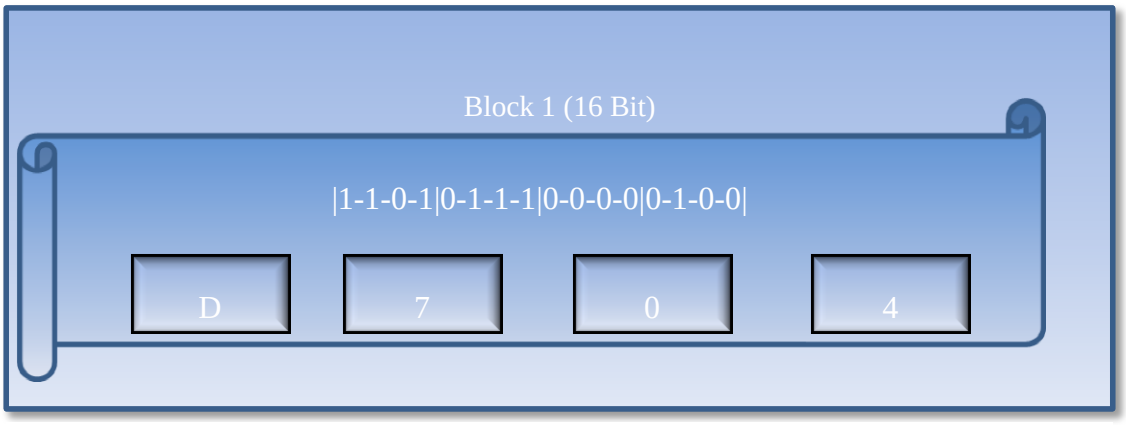


Abbildung 8: RDS Block in Ziffern Beispiel

Dieses Beispiel in Abbildung 8 zeigt den aus 4 Hexadezimalwerten beschrieben Code mit den Wert D704. Welche konkrete Bedeutung hinter diesen Zahlen steht, ist aus folgender Quelle abzulesen. [23] Das D steht für Deutschland, 7 kennzeichnet den Sendebereich als regional, Ziffer 3 mit dem Wert 0 identifiziert Baden-Württemberg und die 4 steht für den Radiosender „Radio Tübingen“. [20]

Vollständigkeitshalber soll an dieser Stelle kurz auf das verwendete Prüfverfahren der Fehlerkorrektur eingegangen werden. Das „Cyclic Redundancy Check“ Verfahren, abgekürzt CRC, mit dessen Hilfe aus einer gegebenen Nachricht ein zugehöriges Codewort konstruiert werden kann, baut auf die Eigenschaften des sogenannten Generatorpolynoms auf. Das Kriterium für die Fehlererkennung besagt, dass das erzeugte Codewort immer dem Vielfachen des Generatorpolynoms entsprechen soll. Wenn dies in der erhaltenden Nachricht nicht der Fall sein sollte, wird dies als fehlerhaft zurück gewiesen. [24] [25]

3.1.4. Dienste

Im Folgenden werden die einzelnen Dienste des RDS erläutert. Dabei stehen Informationsgehalt und Funktionalität im Vordergrund. Wie im Abschnitt 3.1.3. „Aufbau der RDS Nachricht“ beschrieben, wird der Programm-Identifikationscode in jeder Nachricht übertragen. Andere Dienste hingegen können abhängig von der Gruppe sein und werden nachfolgend alphabetisch erklärt.

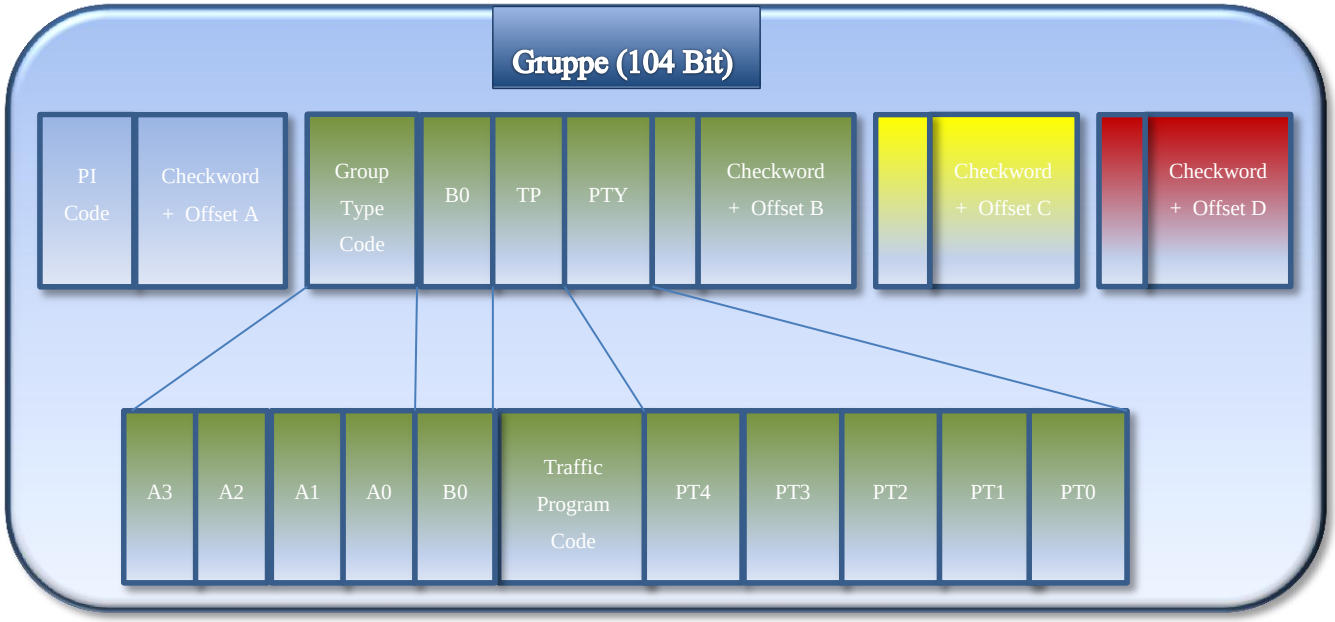


Abbildung 9: RDS Gruppe detailliert

Die Abbildung 9 „RDS Gruppe detailliert“ zeigt den zweiten Block in seiner Unterteilung. Welche Aufgaben die einzelnen Abschnitte haben wird in dem jeweilig zugehörigen Dienst erklärt. [26]

Alternative Frequency (AF)

Dieser Dienst gibt Auskunft über mögliche alternative Frequenzen. So kann dieser genutzt werden, wenn das Signal zu schwach ist, um auf eine andere Frequenz umzuschalten, dies wird in A0 übertragen. [27]

Clock Time (CT)

Dieses Signal dient vor allem der Synchronisation der Empfangsgeräte. Übertragen wird im UTC-Format[8], dessen Größe um den Ortszeitversatz ergänzt wird. Für die Codierung der Stunde werden 24 Möglichkeiten benötigt, dies entspricht $2^5 = 32$, also 5 Bit. Minute und Versatz wird mit je 6 Bits codiert. [27]

[8] Coordinated Universal Time (UTC) beschreibt eine koordinierte Weltzeit dessen Grundformat mit Zeitversatz: 20110525**T142050**+0100 so aussehen kann. Das Datum wäre dann der 25.05.2011 um 13:20:50 Uhr.

Decoder Identification (DI)

Die Decoder Identifikation ermöglicht es, beim Empfänger folgende Einstellungen zu erkennen: [26]

- Mono- oder Stereoempfang
- Künstlicher Programmkopf oder nicht künstlicher Programmkopf
- Komprimiert oder nicht komprimiert
- Statischer PTY-Code oder nicht statischer PTY-Code

Dieser Dienst wird in A0 und B0 übertragen, siehe Abbildung 9 „RDS Gruppe detailliert“.

Enhanced Other Networks (EON)

Dieser Dienst ist entwickelt worden, um große Rundfunknetze mit eigenen Programmen zu stützen. So wird ermöglicht, netzübergreifend auf Informationen zu reagieren. Wenn z.B. im österreichischen Rundfunk Verkehrsnachrichten zu empfangen sind, wird Zeitpunkt und Frequenz mit EON-Daten übertragen. [26]

Music Switch (MS)

Dieser Dienst fungiert wie ein Schalter, welcher genutzt wird, um zu identifizieren, wann Musik oder Sprache übertragen wird. Dieser Dienst wird in A0 und B0 übertragen siehe Abbildung 9 „RDS Gruppe detailliert“. [26]

Open Data Applications (ODA)

Diese Schnittstelle dient der Erweiterbarkeit des RDS, um etwaige nicht berücksichtigte Dienste ohne Anpassung des Standards integrieren zu können. [26]

Program Identification Code (PI-Code)

Der PI-Code wird für die Kennzeichnung des übertragenden Inhalts verwendet. Um Redundanzen in der Arbeit zu vermeiden, soll an dieser Stelle auf Abschnitt 3.1.3. „Aufbau der RDS Nachricht“ verwiesen werden.

Program Service Name (PS)

Der Program Service Name beinhaltet max. 8 alphanumerische Zeichen und gibt Auskunft über den Sendernamen bzw. den Dienstnamen. Der PS wird normalerweise von jedem RDS

Tuner angezeigt. Dabei wird in statischen und dynamischen PS unterschieden. Beim dynamischen PS kann geblättert werden: Dass bedeutet, dass neben dem Programm Service Namen auch noch Liedtitel, Künstler oder längere Stationsnamen angezeigt werden können. Der Program Service Name wird in A0 und B0 übertragen, siehe Abbildung 9 „RDS Gruppe detailliert“. [26]

Program Type (PTY)

Die Programmart ist eine Kennnummer, die von jedem Sender übertragen werden sollte. Sie Identifiziert dabei eine der 31 ihr zu Grunde liegenden Kategorien. Dabei sind z.B. Folgende möglich: Sport, Nachrichten oder Rock-Musik. Dies kann genutzt werden, um nur spezifische Programmarten zu empfangen. Zusätzlich kann der Code für die Stationssuche verwendet werden, da dieser in jeder RDS-Gruppe übertragen wird, und zwar im 2. Block PT0 bis PT4 siehe Abbildung 9 „RDS Gruppe detailliert“. [26]

Radio Paging (RP)

Das Feature wurde entwickelt, um eine Pagerdienstleistung über RDS zu erbringen. Es gibt zwei Radio Paging Protokolle, dass Radio Paging und das Enhanced (verbesserte) Paging Protokoll. Das Radio Paging Protokoll unterstützt einige Mitteilungsarten incl. internationaler Pageranrufe. Ferner ist es möglich einige Dienstleistungen gleichzeitig zu verwenden, um die Pageranrufe zu übertragen und es besitzt einen Energieeinsparungsmodus. Das verbesserte Paging Protokoll unterstützt Mehrfachversorger und Bereichsdienstleistungen. Die Batterieeinsparung ist ebenfalls verbessert worden. Des Weiteren wurde die internationale Funkrufsystemunterstützung implementiert, so dass die Kompatibilität zum RBDS[9] Standard erhöht wurde. In diesem Protokoll können größere Netzmeldungen mit zusätzlichen Mitteilungsarten genutzt werden. [26]

Radio Text (RT)

Dieser Dienst bietet eine weitere Möglichkeit alphanummerische Zeichen zu übertragen an, wobei die Mitteilung aus bis zu 64 Zeichen bestehen kann. Dieser Dienst wird in A2 übertragen, siehe Abbildung 9 „RDS Gruppe detailliert“. [26]

[9] Das Radio Broadcast Data System (RBDS) ist die Bezeichnung für das Amerikanische Radio Data System (RDS).

Traffic Announcement (TA)

Die Verkehrsankündigung wird benutzt, um laufende Verkehrsansagen anzuzeigen. Das Radio kann TA wie folgt nutzen: Bei einer Verkehrsmeldung kann von CD oder USB-Stick auf den Radiosender gewechselt werden und dieser wird gegebenenfalls lauter gestellt. Bei der Endemarkierung wird die Lautstärke wieder verringert und das alte Ausgabemedium gewählt. [26]

Traffic Message Channel (TMC)

Der Traffic Message Channel ist für diese Arbeit der wohl wichtigste Dienst und wird im Abschnitt 3.2. „ Traffic Message Channel“ genauer erklärt. [26]

Traffic Program (TP)

TP wird genutzt, um Stationen zu identifizieren, die Verkehrsprogramme anbieten. Dies geschieht über einen einfachen Flag. Es wird lediglich geschaut, ob TA aktiv ist oder nicht. TP kann wiederum zur automatischen Stationssuche verwendet werden, da es in jeder RDS-Gruppe enthalten ist. Dieser Dienst wird in TP übertragen, siehe Abbildung 9 „RDS Gruppe detailliert“. [26]

Transparent Data Channel (TDC)

Unter der Zuhilfenahme von TDC können beliebige transparente Daten verschickt werden. Hierbei kann der Versorger selbst die Daten definieren. Dabei können bis zu 32 verschiedene Anwendungen adressiert werden. Spezielle RDS Empfänger können hiermit beispielsweise eine pixelbasierende Anzeige bedienen. Die herkömmlichen RDS Empfänger ignorieren die TDC-Daten. [26]

3.2. Traffic Message Channel

Dieser Abschnitt gibt eine Einführung in den Traffic Message Channel und zeigt, wie die Daten empfangen werden können. Ferner wird die Funktionsweise beschrieben und dessen Zusammenspiel mit dem Radio Data System.

3.2.1. Einführung

Der Traffic Message Channel ist ein spezieller Dienst des Radio Data System. Er wird für die Verteilung von Verkehrsdaten in Europa verwendet. Die Verkehrsinformationen werden im Alert-C[10] Standard kodiert abgebildet, dazu mehr in 3.2.3. „Funktionsweise“. [1] Der Empfang ist nur mit entsprechenden Radiogeräten möglich. Wenn der Benutzer durch verschiedene Ländern reist, werden die Mitteilungen immer in der Sprache des jeweiligen Empfangsgerätes angezeigt. Ein weiterer wichtiger Aspekt ist, dass der TMC-Dienst kostenlos für den Nutzer ist. In einigen Ländern gibt es auch die kostenpflichtige Variante TMC-Pro, die eine bessere Qualität der Nachrichten aufweist. Die Einbindung der Pro-Variante ist vom Aufgabensteller DLR aus Kostengründen nicht erwünscht und wird somit im weiteren Verlauf nicht weiter in die Betrachtungen einfließen. [1]

3.2.2. Datenquellen

Der Nutzer erhält einen permanenten Datenstrom der aktuellen Verkehrsmeldungen. Die Sondierung über bereits erhaltende Nachrichten ist jedem Client selbst überlassen. Auf Basis der mitgelieferten Daten kann eine Region oder Fahrstrecke gefiltert werden bzw. können einzelne Meldungen selektiert werden. Die Radiosender, welche TMC in Deutschland zu Verfügung stellen sind folgende:

Deutsche TMC-Sender: (*[28] [29] [30]* Quellen)

- Sender mit TMC-Signal (alphabetisch sortiert):
    - 106.8 Alster Radio (Hamburg)
    - Antenne Aachen
    - Antenne Bayern
    - Antenne Thüringen
    - Bayerischer Rundfunk
    - Deutschlandfunk
    - Deutschlandradio
    - Hessischer Rundfunk
    - Hit-Radio Antenne (Niedersachsen)

[10] Alert-C ist ein europäischer Standard für sprachunabhängigen Austausch von Verkehrsinformationen via RDS-TMC Channel, die Reference ist ISO 14819 [1].

- Hit-Radio Antenne 1 (Baden-Württemberg)
- Hit-Radio FFH (Hessen)
- Mitteldeutscher Rundfunk
- Norddeutscher Rundfunk
- Planet-Radio (Hessen)
- Radio Bremen
- Radio_NRW (Nordrhein-Westfalen)
- RPR1 (Rheinland-Pfalz)
- Saarländischer Rundfunk
- Südwestrundfunk
- Westdeutscher Rundfunk

- Sender in der Region Berlin mit TMC-Signal (nach Frequenzen aufsteigend sortiert):
  - 88.8 Rundfunk Berlin-Brandenburg
  - 95.8 RBB (Radio Berlin/Brandenburg)
  - 99.7 Antenne Brandenburg
  - 104.6 RTL (Berlin/Brandenburg)
  - 105.5 Spreeradio (Berlin/Brandenburg)

Die Sender im Raum Berlin wurden mit dem BT- 465UTE GPS/TMC Empfänger von Navilock überprüft. Dazu wurde mit Hilfe der GNS-API ein Userinterface programmiert, das dies ermöglicht.

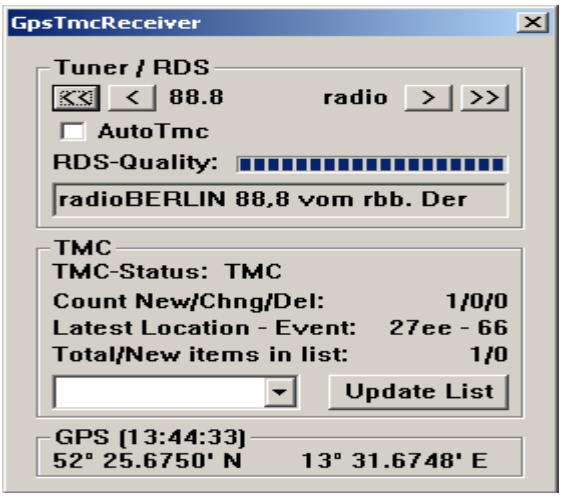


Abbildung 10: GpsTmcReceiver

Die Abbildung 10 “GpsTmcReceiver“ zeigt das Interface, mit dem die Radiosender mit TMC gefunden wurden. Innerhalb der Ansicht gibt es drei Bereiche: der Tuner/RDS, TMC und der GPS[11] Ausschnitt. Im Tuner/RDS-Bereich gibt es die Möglichkeit, die zu überprüfenden Frequenzen einzustellen. Der einfache Pfeil verändert die Frequenz im 0.1 MHz Bereich, der Doppelfeil erhöht bzw. verringert die Frequenz in 0.1 MHz Schritten so lange, bis ein RDS-Signal gefunden wurde, dies wird mit dem blauen Balken angezeigt. Die Skala hat einen

[11] Global Positioning System (GPS) ist ein globales Satellitennavigationssystem. [5][S23]

Wertebereich von 0 = kein Balken, bis 10 = voller Balken. In dem darunterliegenden Textfeld wird der RDS-String angezeigt, „radioBERLIN 88,8 vom rbb. Der …“, den der jeweilige Radiosender ausstrahlt. Wenn ein Haken bei „AutoTmc“ gesetzt wird, sucht das Programm solange nach einem Radiosender, bis dieser den Flag bei „Traffic Program (TP)“ entsprechend findet und somit weiß, dass dieser Verkehrsmeldungen ausstrahlt. In dem Bereich TMC wird nun die empfangene Nachricht angezeigt. Des Weiteren wird der Location-Code angezeigt und das entsprechende Event dazu. Was diese beiden Werte aussagen, wird unter Abschnitt 3.2.3. „Funktionsweise“ näher erläutert. Der GPS Bereich zeigt, wo die Verkehrsnachricht empfangen wurde, in diesem Fall in Berlin Adlershof bzw. bei 52.256750 nördlicher Breite und 13.316748 östlicher Länge.

Eine TMC-Meldung ist nicht das Resultat eines Verkehrserfassungssystems, vielmehr gibt es verschiedene Wege, auf denen eine mögliche Meldung entstehen kann. Im Folgenden werden die gängigsten Möglichkeiten vorgestellt.

Ein mögliches Szenario beschreibt den Weg über die örtlichen Sicherungsorgane, wie die Polizei. Eine Polizeistelle erhält Kenntnis über eine Verkehrsstörung, mit oder ohne Überprüfung geht diese Information an die jeweilige Landesmeldestelle der Polizei. Diese senden die Meldung an einen Server in Düsseldorf. Dieser zentrale Knotenpunkt garantiert die Verbreitung im gesamten Bundesgebiet. Einige Radiosender, wie z.B. „Rundfunk Berlin Brandenburg“ (rbb) haben Zugriff auf diesen Server. Beim Radiosender selbst können weitere Informationen hinzukommen, wie Hörermeldungen oder andere zugängliche Verkehrserfassungssysteme bzw. die daraus generierten Informationen. Ein gutes Beispiel ist der Radiosender rbb, welcher in der Verkehrsmanagementzentrale (VMZ) Berlin in Tempelhof sitzt. Das hiesige Erfassungssystem der VMZ generiert Verkehrsinformationen auf Basis von Induktionsschleifen, Videosystemen und zum Teil der vom DLR vorprozessierten FC-Daten. Dabei ist zu bedenken, dass dem ansässigen Radiosender in der VMZ nicht alle Daten zur Verfügung stehen. Die folgende Abbildung soll dies weiter verdeutlichen. [31]

Abbildung 11: TMC Übertragungskette [31]

Die Abbildung 11 „TMC Übertragungskette“ zeigt den zuvor beschriebenen Weg der Verkehrsmeldung. Am Punkt 1 wird durch ein geeignetes Erfassungsgerät die Störung erkannt, in dieser Abbildung eine Kamera. Denkbar ist ebenfalls, wie bereits beschrieben, der eintreffende Polizeibeamte bzw. der abgesetzte Notruf an Krankenwagen oder Feuerwehr. Diese Meldung wird an einen zentralen Punkt weiter gegeben (Punkt 2). Dieser Punkt kann sowohl der zentrale Server eines Bundesorgans sein, wie der in Düsseldorf, oder direkt das hiesige Verkehrskontrollzentrum, wie die VMZ in Berlin. Die Radiosender (Punkt 3) beziehen ihre Informationen von diesen Stellen und geben diese an alle TMC-Empfänger weiter, (Fahrzeug an Punkt 4). Dieses kann auf Basis der neuen Informationen eine alternative Route wählen (gelbe Pfeile ab Punkt 5). [31]

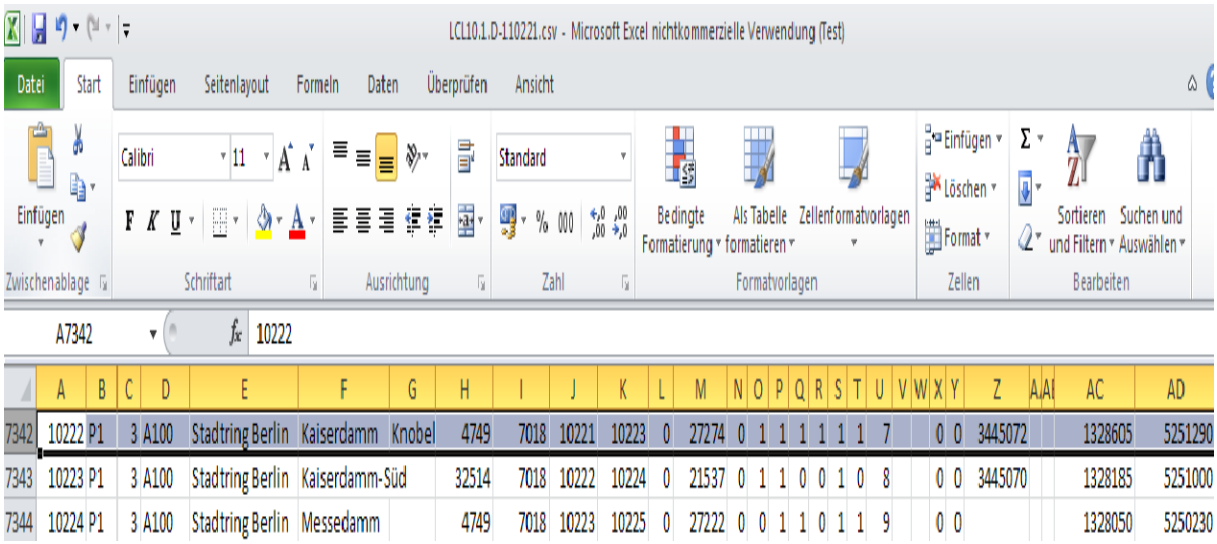
Zusammenfassend kann man auf Basis der unterschiedlichen Informationsquellen prognostizieren, dass die Meldungen unterschiedlicher Service Provider in ihrer Qualität stark voneinander abweichen werden. Eine genauere Analyse der Daten ist unter Kapitel 6. „TMC Datenanalyse“ nachzulesen.

3.2.3. Funktionsweise

Jede TMC-Meldung ist eine eigenständige Nachricht, die folgenden Informationsgehalt tragen kann. Auch hier soll der EN ISO 14819 Standard als Quelle dienen. [1]

Abbildung 12: GpsTmcReceiver Markiert

Jede Nachricht besitzt zwei wesentliche Informationen: den Location-Code und den Event-Code, in der Abbildung 12 „ GpsTmcReceiver Markiert“ rot hervorgehoben. Der Location-Code (27EE) spezifiziert dabei den Ort des durch den Event-Code (66) verschlüsselten, Ereignisses. Die Verwaltung und Herausgabe dieser beiden Tabellen obliegt der Bundesanstalt für Straßenwesen, abgekürzt BASt. Auf der Seite der BASt ist es möglich, die aktuellen Decodierungstabellen zu erhalten. [32] Die folgende Abbildung zeigt einen Ausschnitt aus der aktuellen Location-Code-List (LCL).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD
7342	10222	P1		3	A100	Stadtring Berlin	Kaiserdamm	Knobel	4749	7018	10221	10223	0	27274	0	1	1	1	1	1	7			0	0	3445072			1328605	5251290
7343	10223	P1		3	A100	Stadtring Berlin	Kaiserdamm-Süd		32514	7018	10222	10224	0	21537	0	1	1	0	0	1	0	8			0	0	3445070			1328185	5251000
7344	10224	P1		3	A100	Stadtring Berlin	Messedamm		4749	7018	10223	10225	0	27222	0	0	1	1	0	1	1	9			0	0				1328050	5250230

Abbildung 13: Location-Code-List [32]

Die in der Abbildung 13 markierte Zeile bezieht sich auf die in Abbildung 12 „ GpsTmcReceiver Markiert“ empfange Nachricht mit dem Location-Code 27EE und dem Event-Code 66 als Hexadezimalwerte. Die in der Tabelle dargestellten Werte beziehen sich auf die Dezimalwerte von umgerechnet 10222 (Spalte A, Location-Code) und 102 (Event-

Code). Der umgerechnete Wert gibt somit die Zeile vor, aus der die folgenden Informationen abgeleitet werden können:

Tabelle 1: Location-Code-List Beispiel [32]

Spalt	Bedeutung	Beispiel	Spalt	Bedeutung	Beispiel
A	LOCATIONCODE	10222	**S**	PRESENT_POSITIVE	1
B	TYPE	P1	**T**	PRESENT_NEGATIVE	1
C	SUBTYPE	3	**U**	EXIT_NUMBER	7
D	ROADNUMBER	A100	**V**	DIVERSION_POSITIVE	
E	ROADNAME	Stadtring Berlin	**W**	DIVERSION_NEGATIV	
F	FIRST_NAME	Kasierdamm	**X**	CHANGE	0
G	SECOND_NAME	Knobelsdorffstraß	**Y**	TERN	0
H	AREA_REFERENCE	4749	**Z**	NETZKNOTEN_NR	344507
I	LINEAR_REFERENC	7018	**AA**	NETZKNOTEN2_NR	
J	NEGATIVE_OFFSET	10221	**AB**	STATION	
K	POSITIVE_OFFSET	10223	**AC**	X_KOORD	132860
L	URBAN	0	**AD**	Y_KOORD	525129
M	INTERSECTIONCOD	27274	**AE**	POLDIR	A24
N	INTERRUPTS_ROAD	0	**AF**	ADMIN_County	D.BE
O	IN_POSITIVE	1	**AG**	ACTUALITY	
P	OUT_POSITIVE	1	**AH**	ACTIVATED	1
Q	IN_NEGATIVE	1	**AI**	TESTED	
R	OUT_NEGATIVE	1	**AJ-AS**	SPECIAL1- SPECIAL10	

Aus der Tabelle 1 „Location-Code-List Beispiel“ können die spezifischen Werte für diesen Location-Code entnommen werden. Wichtige Werte sind in der Spalte AC und AD zu finden, welche die X-Y-Koordinaten enthalten.

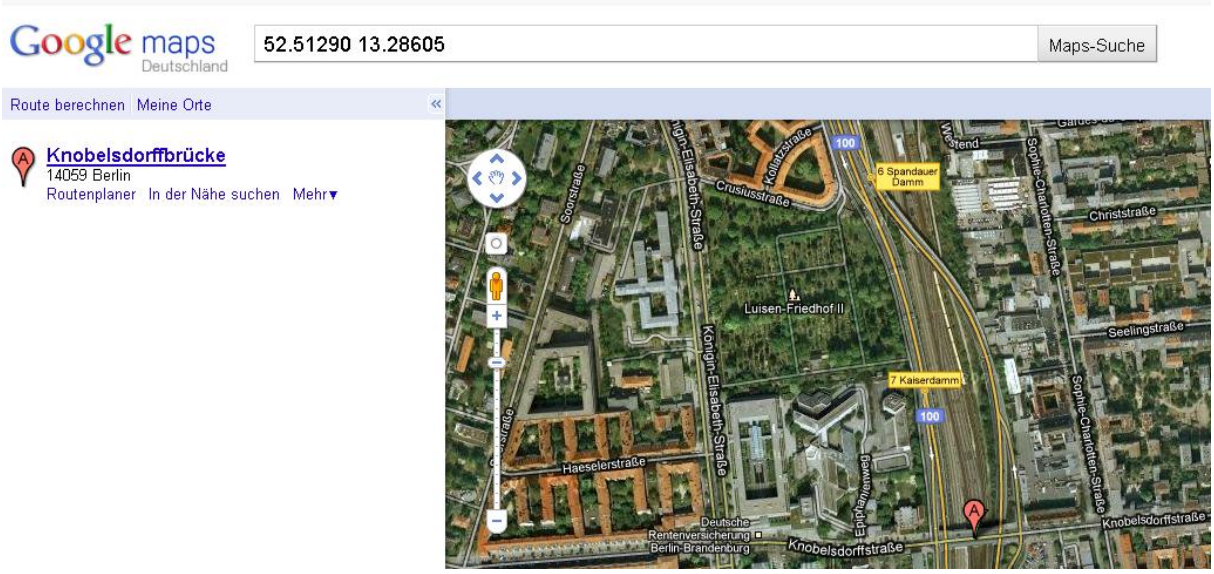
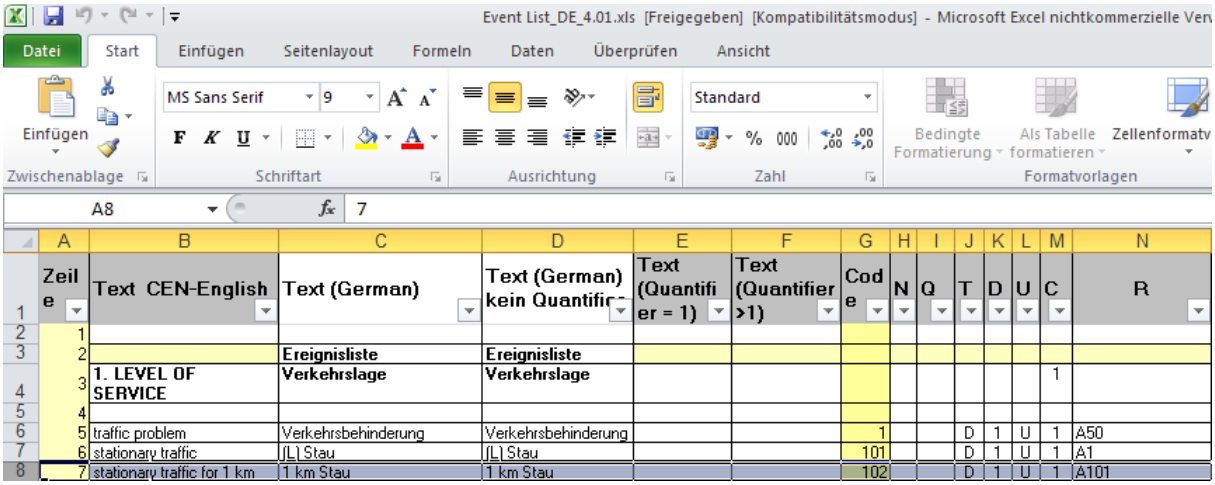


Abbildung 14: Lat & Lon LCL Beispiel [33]

Die Abbildung 14 „Lat & Lon LCL Beispiel“ zeigt, wo sich diese Koordinaten laut Google-Maps befinden. Des Weiteren ist der Name „Kaiserdamm“ aus Spalte F zu entnehmen und die Straßennummer (Spalte D, A100), Straßenname (Spalte E, Stadtring Berlin) sowie der Typ der Meldung (Spalte B, P1), wobei P für Point steht und die Zahl für die Anzahl der Punkte. [32] Um die Funktionsweise der TMC-Meldungen zu verstehen, ist es nicht nötig jede Bedeutung an dieser Stelle zu erklären. Ein genaueres Bild wird im Kapitel 5 „Empfangen der TMC-Meldungen“ gegeben.

Wichtig ist, dass auf Basis der Location-Code-List relevante Informationen über Ort, Name, Ausprägung, Richtung und Gebiet des jeweiligen Ereignisses gegeben werden. Welches Ereignis an dieser Stelle vorliegt, ist aus der Event-List zu entnehmen.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Zeile	Text CEN-English	Text (German)	Text (German) kein Quantifi	Text (Quantifier = 1)	Text (Quantifier >1)	Code	N	Q	T	D	U	C	R
2	1													
3	2		Ereignisliste	Ereignisliste										
4	3	1. LEVEL OF SERVICE	Verkehrslage	Verkehrslage									1	
5	4													
6	5	traffic problem	Verkehrsbehinderung	Verkehrsbehinderung			1			D	1	U	1	A50
7	6	stationary traffic	/L\ Stau	/L\ Stau			101			D	1	U	1	A1
8	7	stationary traffic for 1 km	1 km Stau	1 km Stau			102			D	1	U	1	A101

Abbildung 15: Event-List [32]

Die Abbildung 15 „Event-List“ zeigt das bekannte Beispiel aus Abbildung 12 für den Event-Code. Auch hier soll diese Abbildung tabellarisch abgebildet werden, um einen besseren Überblick zu verschaffen.

Tabelle 2: Event-List Beispiel [32]

Spalte	Bedeutung	Beispiel	Spalte	Bedeutung	Beispiel
A	Zeile	7	**J**	T	D
B	Text CEN-English	stationary traffic for 1 km	**H**	N	
C	Text (German)	1 km Stau	**I**	Q	
D	Text (German) kein Quantifier	1 km Stau	**J**	T	D
E	Text (Quantifier = 1)		**K**	D	1
F	Text (Quantifier > 1)		**L**	U	U
G	Code	102	**M**	C	1
H	N		**N**	R	A101
I	Q				

Wie bereits in Tabelle 1 „Location-Code-List Beispiel“ zu sehen ist, enthält nicht jede Zeile in jeder Spalte Werte. Dies resultiert aus der Tatsache, dass es nicht für jedes Ereignis bzw. jeden Ort Zusatzinformationen gibt. Das Event 102 ist in der Tabelle 2 „Event-List Beispiel “ herausgestellt. Spalte G enthält hier die Referenz, also den eindeutigen Code mit dem Wert 102. Ferner ist aus der Tabelle abzulesen, dass dieses Event keine Quantifier Zusatzinformationen enthält, Spalte D „… kein Quantifier“ und das ein Stau in der Länge von 1 km vor Ort zu erwarten ist. Für die Spalte H mit dem Kürzel N gibt es drei Möglichkeiten für Werte. Ein Blank, also leeres Feld, weist darauf hin, dass es sich um eine Information handelt. Bei einem großen F (forecast) handelt sich um eine Prognose. Die dritte zu erwartende Größe ist ein S (silent), eine sogenannte ruhige Nachricht. Diese wird im

Empfänger oft nicht angezeigt, da sie gefiltert ist. Meist sind Meldungen über Aufhebungen von Begrenzungen mit dieser Zusatzinformation versehen. Die Spalte I beinhaltet den Quantifier hier mit einem Q abgekürzt. Dieser Wert ist optional und referenziert die folgende Tabelle. [34] [35] [36]

Tabelle 3: Quantifier-List [32]

	Label 2	Label 3	Label 4	Label 4	Label 4	Label 4	Label 4	Label 4
			Quantifier 0	Quantifier 1	Quantifier 2	Quantifier 3	Quantifier 4	Quantifier 5
Code	Betroffener Abschnitt	Geschwindigkeits-beschränkung	Kleine Zahlen	Zahlen	Sichtweiten	Wahrscheinlichkeit	Geschwindigkeit	Dauer
1	1 km	5 km	1	1	unter 10 m	5 %	5 km/h	bis zu 5 Minuten
2	2 km	10 km	2	2	unter 20 m	10 %	10 km/h	bis zu 10 Minuten
3	3 km	15 km	3	3	unter 30 m	15 %	15 km/h	bis zu 15 Minuten
4	4 km	20 km	4	4	unter 40 m	20 %	20 km/h	bis zu 20 Minuten
5	5 km	25 km	5	10	unter 50 m	25 %	25 km/h	bis zu 25 Minuten
6	6 km	30 km	6	20	unter 60 m	30 %	30 km/h	bis zu 30 Minuten
7	7 km	35 km	7	30	unter 70 m	35 %	35 km/h	bis zu 35 Minuten
8	8 km	40 km	8	40	unter 80 m	40 %	40 km/h	bis zu 40 Minuten
9	9 km	45 km	9	50	unter 90 m	45 %	45 km/h	bis zu 45 Minuten
10	10 km	50 km	10	60	unter 100 m	50 %	50 km/h	bis zu 50 Minuten
11	12 km	55 km	11	70	unter 110 m	55 %	55 km/h	bis zu 1 Stunden
12	14 km	60 km	12	80	unter 120 m	60 %	60 km/h	bis zu 2 Stunden
13	16 km	65 km	13	90	unter 130 m	65 %	65 km/h	bis zu 3 Stunden
14	18 km	70 km	14	100	unter 140 m	70 %	70 km/h	bis zu 4 Stunden
15	20 km	75 km	15	150	unter 150 m	75 %	75 km/h	bis zu 5 Stunden
16	25 km	80 km	16	200	unter 160 m	80 %	80 km/h	bis zu 6 Stunden
17	30 km	85 km	17	250	unter 170 m	85 %	85 km/h	bis zu 7 Stunden
18	35 km	90 km	18	300	unter 180 m	90 %	90 km/h	bis zu 8 Stunden
19	40 km	95 km	19	350	unter 190 m	95 %	95 km/h	bis zu 9 Stunden
20	45 km	100 km	20	400	unter 200 m	100 %	100 km/h	bis zu 10 Stunden
21	50 km	105 km	21	450	unter 210 m	-	105 km/h	bis zu 11 Stunden
22	55 km	110 km	22	500	unter 220 m	-	110 km/h	bis zu 12 Stunden
23	60 km	115 km	23	550	unter 230 m	-	115 km/h	bis zu 18 Stunden
24	65 km	120 km	24	600	unter 240 m	-	120 km/h	bis zu 24 Stunden
25	70 km	125 km	25	650	unter 250 m	-	125 km/h	bis zu 30 Stunden
26	75 km	130 km	26	700	unter 260 m	-	130 km/h	bis zu 36 Stunden
27	80 km	-	27	750	unter 270 m	-	135 km/h	bis zu 42 Stunden
28	85 km	-	28	800	unter 280 m	-	140 km/h	bis zu 48 Stunden

29	90 km	-	30	850	unter 290 m	-	145 km/h	bis zu 54 Stunden
30	95 km	-	32	900	unter 300 m	-	150 km/h	bis zu 60 Stunden
31	100 km	-	34	950	-	-	155 km/h	bis zu 66 Stunden
0 (=32)	> 100 km	-	36	1000	-	0%	160 km/h	bis zu 72 Stunden

Wenn ein Quantifier gegeben ist, können in der Tabelle 3 „Quantifier-List“ mit dem entsprechenden Code Zusatzinformationen zu dem jeweiligen Event ermittelt werden. So würde beim Quantifier-Code 4, zum Beispiel der betroffene Streckabschnitt eine Länge von 4 km haben und eine Geschwindigkeitsbeschränkung von 20 km/h, ferner ist eine Passierdauer von 20 min zu erwarten.

Die Spalte J beinhaltet eine großes T für Time oder nach ISO 14819-2 Standard [32] auch Duration, also Dauer. Dabei wird nicht die genaue Zeit bis zum Auflösen eines Staus oder einer Baustelle gegeben, da dies nach heutigen Betrachtungen nur unter Zuhilfenahme einer enormen Energiemenge überhaupt möglich wäre, [37] sondern es wird dem Nutzer eine Abschätzung mitgeteilt. Ein großes D für Dynamisch deutet dabei auf eine zu erwartende kurze Dauer hin, wobei ein L ein länger anhaltendes Ereignis prognostiziert. Die Spalte K gibt mit groß D die Richtung vor. Dabei steht eine 1 für nur eine Richtung des Straßenverkehrs und eine 2 für beide Richtungen der Straße. Spalte L gibt Auskunft über die Dringlichkeit (urgency) abgekürzt mit einem U. Der Wert ist entweder U = dringend, X = extrem dringend oder leer = diese Information liegt nicht vor. Die Updatekategorie wird in der Spalte M mit dem Kürzel C dargestellt, welches genutzt wird, um die Aktualisierung zu regulieren. Die 1 in dem Beispiel weist darauf hin, dass es zu dem dazugehörigen Location-Code keine weiteren Ereignisse gibt, also nur dieses eine. Bei einer 2 hingegen würde ein zweites Event die Situation vor Ort näher beschreiben, z.B. das Event mit dem Code 928 für Hochwasser. Jedem Ort können dabei maximal 5 Ereignisse zugeordnet werden. Die letzte Spalte N kann eine weitere Reference enthalten, abgekürzt durch ein R. Dies wird mit einem Kennbuchstaben (A - Z) und einer Kennziffer (1 - 999) in Kombination dargestellt. Die Reference A101 wird wie folgt interpretiert: A steht für „Level of Service“, die Zahl 101 für Stau. Wie diese Codes zu analysieren sind, kann im ISO 14819-2 Standard unter Abschnitt 3 „Event and Information codes for Traffic Message Channel“ nachgelesen werden. [32] [34]

Die Zusammenführung der Location-Code-List und der Event-List geben Auskunft über das jeweilige Ereignis und an welchem Ort bzw. in welcher Region dies stattfindet. Die Location-Code-List für den Raum Deutschland enthält 44234 Zeilen und somit referenzzierbare Punkte

auf der Karte. Die Event-List enthält 1615 unterschiedliche Ereignisse. Diese lassen sich grob in den Kategorien Unfälle, Staus, Sperrungen, Wettereinflüsse und Geschwindigkeitsregularien unterscheiden. Jede TMC-Meldung ist in dem Alert-C Standard kodiert. Somit würde das Ereignis der TMC-Meldung im Beispiel, in Frankreich oder England denselben Event-Code tragen. Die Location-Code-List hingegen pflegt jedes Land selbst, in Deutschland ist dafür die BASt zuständig.

3.3. Floating Car Data

In diesem Abschnitt wird das Floating Car Data (FCD) Konzept vorgestellt. Auch hier werden Grundlagen vermittelt, die unabdingbar sind, um die spätere Fusion der Daten und den Aufbau einer neuen DLR-Prozessierungskette zu verstehen.

3.3.1. Einführung

Die Floating Car Data Technologie beschreibt ein Verfahren, bei dem Verkehrsteilnehmer, ihre GPS-Position mit Länge und Breite sowie Zeitstempel, Fahrzeug-ID, Status und Momentangeschwindigkeit, an einen zentralen Ort senden, wo die Daten zu Verkehrsinformationen weiterverarbeitet werden. Das DLR ist seit 2001 an der Nutzung und Erforschung dieser Technologie beteiligt und stellt mit seinen langjährigen Erfahrungen die Kernkompetenz in diesem Bereich in Europa dar. [38] Dieses dynamische Erfassungsverfahren beinhaltet einige Vorteile gegenüber einem lokalen Verfahren. Selbst im untergeordneten Verkehrsnetz, wo lokale Detektion viel zu teuer ist, kann FCD Verkehrsinformationen liefern. Bei einem lokalen Verfahren, wie einer Kamera hingegen, wird ein auftretendes Verkehrsdefizit nur an einer Stelle erkannt. Positiv ist, dass dessen Bestimmung im Falle einer Störung sicher ist, wenn das entsprechende Gerät funktioniert. Des Weiteren stellt die FCD-Lösung eine schnelle und kostengünstige Variante dar, da ein FCD-Gerät nur wenige hundert Euro kostet. Ein vergleichbares lokales System hingegen kostet mehrere tausend Euro, wobei der Bezugswert sich auf die Kosten für ein Erfassungsgerät bezieht. [38] Ferner ist dieses System schnell und einfach für andere Städte erweiterbar.

3.3.2. Datenerhebung

Die Datenerhebung beim DLR erfolgt mit Hilfe der hiesigen Taxiunternehmen bzw. deren jeweilige Flotte. Der Vorteil ist, dass jede größere Stadt eine oder mehrere Taxizentralen besitzt. Des Weiteren werden Taxis öfter bewegt als beispielsweise privat genutzte Fahrzeuge, somit können mehr Daten für einen größeren Bereich erfasst werden. Ferner gibt es für Probleme im System oder mit der Technik einen Ansprechpartner, die entsprechende Zentrale. [38]

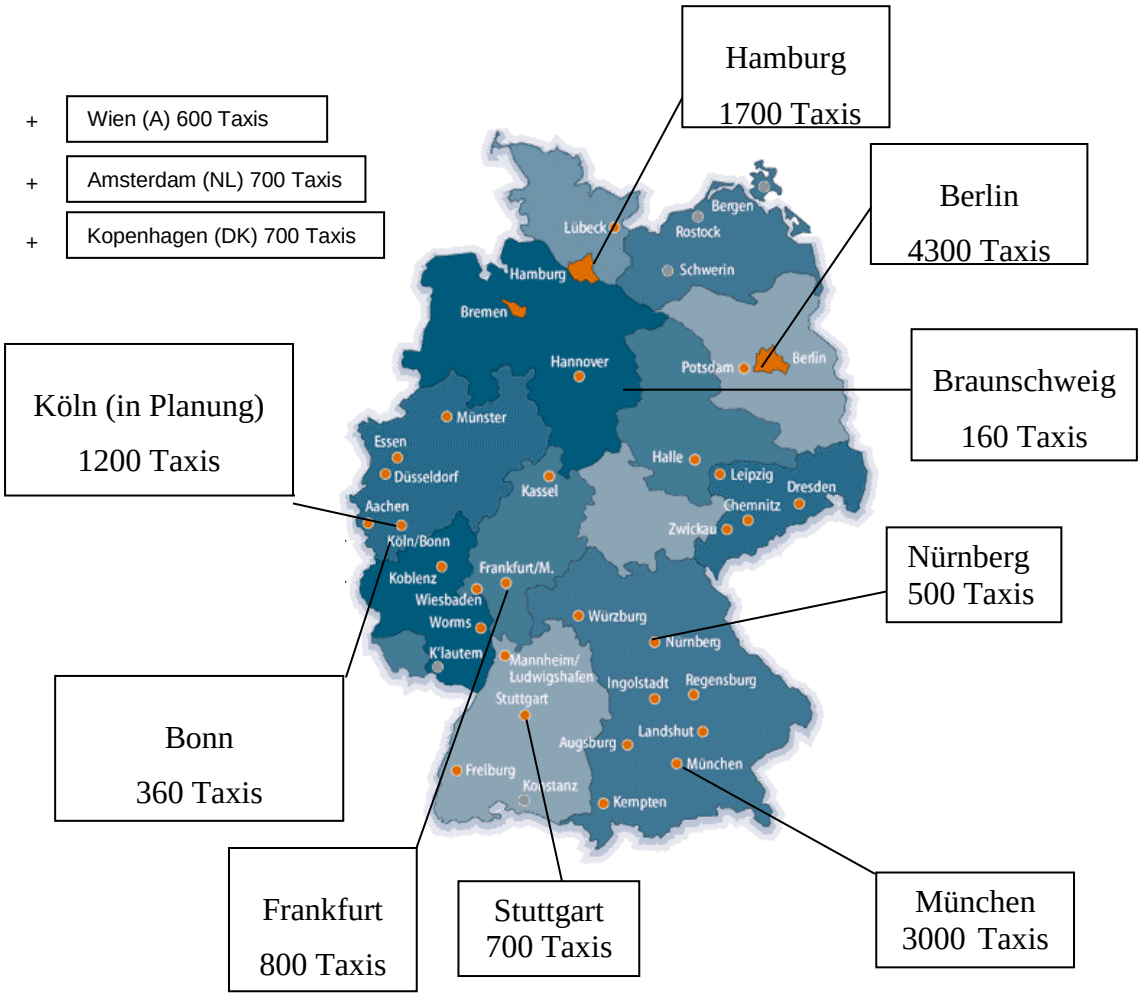


Abbildung 16: FCD Datenerfassung

Die Abbildung 16 „FCD Datenerfassung“ zeigt die Datenquellen des DLR in der Bundesrepublik. Dieses Echtzeitsystem basiert ausschließlich auf Taxis bzw. der Kommunikation mit der dazugehörigen Zentrale. Das gesamte System erfasst mehr als 10.000 FCD-Fahrzeuge, dabei versenden diese im Schnitt 288 Millionen Positionsdaten mit den entsprechenden Zusatzinformationen im Monat, für den Bereich Berlin sind es immer noch rund 30 Millionen FC-Daten. Die einfache Übertragungsmöglichkeit des Systems führte zu weiteren Quellen in ganz Europa, wie Wien, Amsterdam und Kopenhagen. Wie die Daten übertragen werden, soll im Folgenden Abschnitt beschrieben werden. [38]

3.3.3. Funktionsweise

Die beim DLR vorliegende Verarbeitungskette der Daten vom Taxi bis hin zum Nutzer wird im groben in der folgenden Abbildung veranschaulicht.

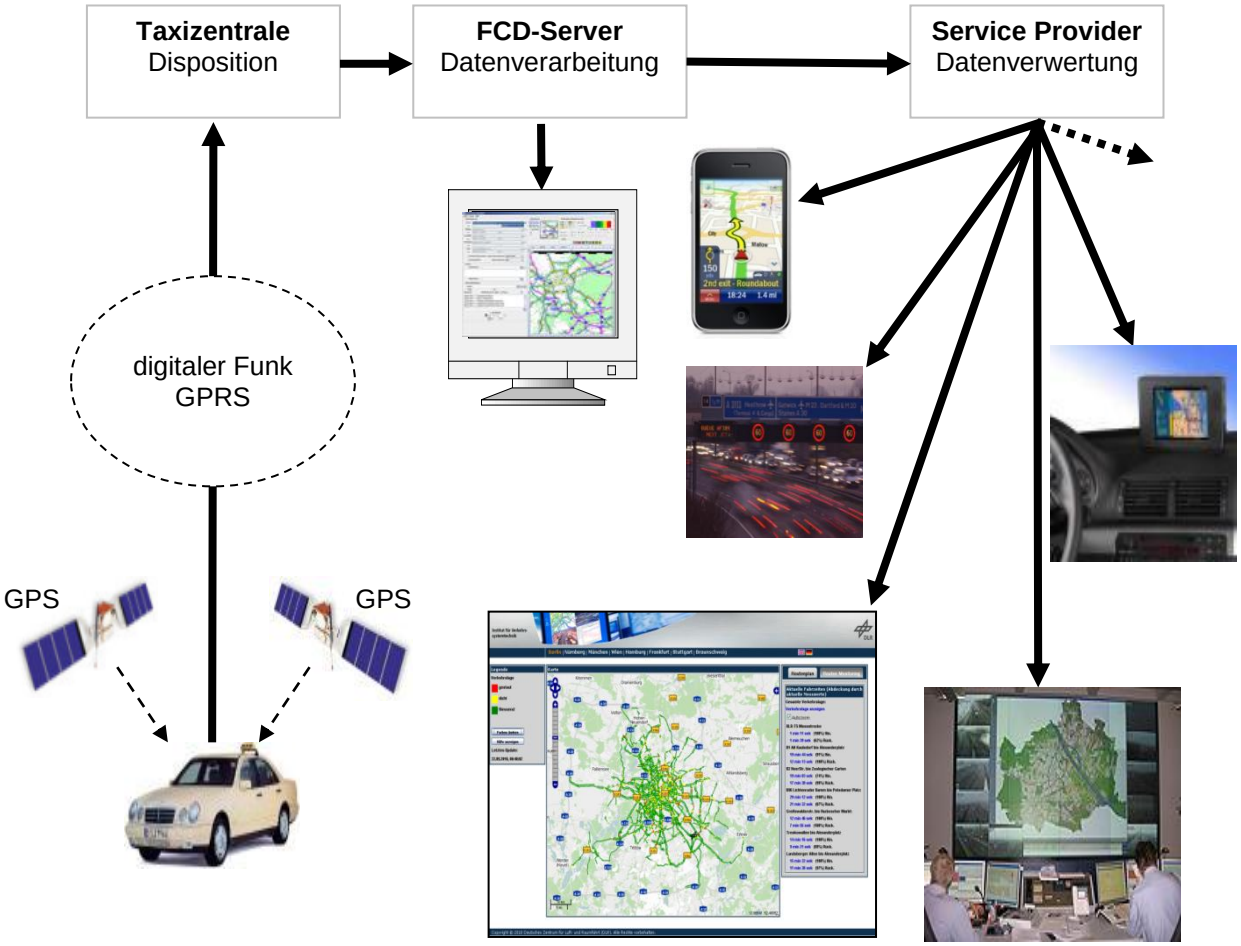


Abbildung 17: FCD Funktionsweise [38]

Die Abbildung 17 „FCD Funktionsweise“ hat ihren Anfang beim Taxi, unten links. Jedes dieser Fahrzeuge besitzt einen GPS-Empfänger und das entsprechende FCD-Modul. Das GPS-Gerät bestimmt dabei die Position in Länge und Breite sowie die Geschwindigkeit. Das Gerät versieht diese Informationen mit Zeitstempel, Fahrzeug-ID und Status.

Tabelle 4: FCD Status [38]

	Austrosoft	Gefos	DLR	Erläuterung
FREE	70	10	20	frei in umgebung
REGISTERED (LOGGED)	65	11	30	angemeldet
WAITING	87	14,15	40	Pause, wartend beim Kunden
OCCUPIED	66.79.90	30.31.32 .34.38	50	besetzt mit Kunde/Fahrziel/Folgeauftrag/besetzt gemeldet
APPROACH (JOURNEY)	75	16	60	"ohne Kunde" Anfahrt zum Kunden, Zielgemeldet
STAND (HALT)	83	17-20	70	Halteplatz, von Zentrale am HP eingebucht

Die Tabelle 4 „FCD Status“ zeigt die Statuscodes der FCD Lieferanten Austrosoft und GefoS. So würde Austrosoft Status = 87 übertragen, wenn das Taxi eine Pause macht. Bei GefoS wird zusätzlich zwischen „Pause“ oder „wartend beim Kunden“ unterschieden. Alle drei Werte werden beim DLR mit dem Wert 40 übersetzt, um eine einheitliche Größe zur Weiterverarbeitung zu schaffen. [18]

Die aufgenommenen Daten sendet das Fahrzeug über GPRS[12] an die zugehörige Taxizentrale. Diese stellen die Daten über einen Service bereit und das DLR holt sich diese zur Prozessierung. Wie diese Verarbeitung intern geschieht, wird im Abschnitt 4.1. „ Istzustand“ näher erklärt. Die so erstellte Verkehrslage für den entsprechenden Bereich kann über das Web Frontend den „City Router“ (Karte rechts neben dem Taxi siehe Abbildung 17) abgerufen werden. Ferner sollen die Daten auch mobilen Nutzern zugänglich gemacht werden. Die Umsetzung erfolgt in einer weiteren Masterarbeit am Standort. Die Verkehrslage kann ebenfalls Empfangsgeräten in Fahrzeugen zur Verfügung gestellt werden. Ferner werden die Daten für den Raum Berlin in sondierter Form an die VMZ übertragen.

[12] General Packet Radio Service (GPRS) ist ein Dienst zur Datenübertragung in UMTS oder GSM. UMTS oder Universal Mobile Telecommunications System ist ein Mobilfunkstandard der dritten Generation auch 3G. GSM (Global System for Mobile Communications) ist ebenfalls ein Standard für volldigitale Mobilfunknetze der sogenannten zweiten Generation (2G).

4. Anforderungsanalyse

Die Anforderungsanalyse umfasst den momentanen Status der FC-Daten-Prozessierung, vom Empfang über die Verarbeitung bis hin zur Darstellung der im System befindlichen Daten. Im Folgenden ist der Soll-Zustand zu definieren sowie zu klären, was erreicht werden soll und welche Mittel sich aus welchen Gründen anbieten. In diesem Kapitel wird ebenfalls das Lösungskonzept vorgestellt, welches einen Überblick über das zu erreichende Endprodukt veranschaulichen soll. Des Weiteren ist die Abbildung 18 der rote Faden durch dieses Kapitel und soll vollständig erklärt werden.

4.1. Istzustand

Der Status Quo[13] der FCD-Kette beschreibt modulares System, dessen wesentliche Komponenten in der folgenden Abbildung veranschaulicht werden sollen.

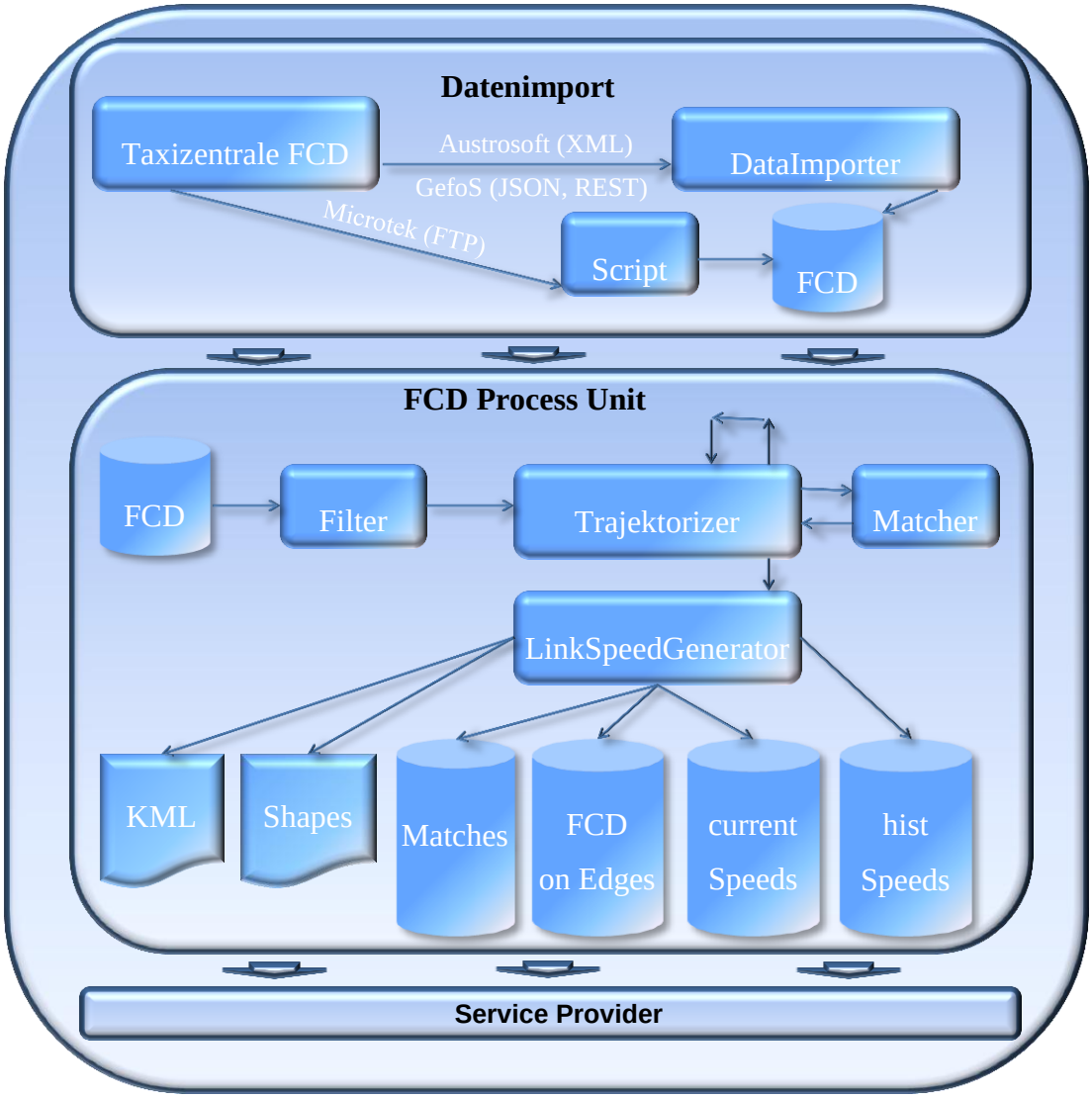


Abbildung 18: FCD-Kette

[13] Status Quo lateinisch für den aktuellen Zustand des im kontext spezifizierten Mediums.

4.1.1. Datenprovider & Datenimport

Die Abbildung 18 „FCD-Kette“ knüpft an die im Abschnitt 3.3. „ Floating Car Data“ beschriebenen Grundlagen an. Die Taxizentralen speichern die FCD-Daten ihrer Flotte und der DLR-DataImporter holt sich diese über einen bereitgestellten Service. Dieses Modul schreibt die FC-Daten in eine PostgreSQL Datenbank. PostgreSQL ist ein objektrelationales Datenbankmanagementsystem und kann mit Objekten arbeiten. Bei früheren relationalen Datenbanken, wie die Alegro-C Datenbank aus den 90iger Jahren, konnten komplexe Objekte nicht gespeichert werden bzw. es musste zu deren Speicherung eine Tabellenstruktur geschaffen werden, die es ermöglichte, das Objekt mit seinen Werten zu speichern. Das aus der Programmierung bekannte Objekt moduliert dabei beispielsweise ein reales Objekt, wie z.B. ein Auto. Dieses Auto wird durch verschiedenste Merkmale gekennzeichnet, wie Maximalgeschwindigkeit, Hubraum, Gewicht oder Anzahl der Türen. Dieses Objekt kann in objektrelationalen Datenbankmanagementsystemen als Ganzes abgebildet bzw. zurückgegeben werden. Die dazu nötigen Strukturen werden intern von der Datenbank selbst aufgebaut und verwaltet. Ausgehend von dem FCD-Objekt soll im Folgenden die Verarbeitung der Rohdaten beschrieben werden. [39]

Die von den Taxizentralen gelieferten Rohdaten werden in die Tabelle „tdp_fcd_raw“[14] der entsprechenden Stadt gespeichert, wie die Abbildung 19 zeigt.

Daten editieren - PostgreSQL 8.4 (129.247.218.180:5432) - tdp_bonn - tdp_fcd_raw

Datei Bearbeiten Anzeigen Werkzeuge Hilfe

100 Zeilen.

	veh_id [PK] integer	local_time [PK] timestamp without ti	position geography(Point,4326)	status smallint	speed smallint	angle smallint
1	100010	2011-06-14 06:00:22	0101000020E6100000000000002055621C400000000033604940	0		
2	100010	2011-06-14 06:00:47	0101000020E610000000000060DE631C4000000080FB5F4940	0		
3	100010	2011-06-14 06:02:58	0101000020E6100000000000806164 1C40000000C0E35F4940	0		

Abbildung 19: FCD-Raw

Die Spalte „veh_id“ referenziert hierbei eineindeutig jede unternommene Fahrt eines Taxis, nicht das Auto selbst. Dies ist notwendig, um die Anonymität eines jeden Taxis sicherzustellen, wie es mit den Taxizentralen vereinbart ist. Die Vehicle-ID wird bei jeder Fahrt neu vergeben. In Abbildung 19 „FCD-Raw“ sind drei Datensätze abgebildet, die alle zur selben Tour gehören. Diese sind mit einem Zeitstempel versehen, der durch die Spalte

[14] Traffic Data Plattform (tdp) ist der Sammelbegriff für alle Daten und Datenbanken des DLR, die im Kontakt mit FCD oder TMC stehen.

„local_time“ beschrieben wird. Wo diese Daten aufgenommen wurden, ist unter „position“ abzulesen. Die hier enthaltenen Werte entsprechen dem PostGIS-Format[15] für einen Punkt und können als Objekte verwaltet werden. Der genaue interne Aufbau des Formates ist für dessen Arbeit nicht wichtig, da über bereitgestellte Funktionen z.B. folgende Abfrage gestellt werden kann, um Längen(x)- und Breitengrad(y)-Koordinaten aus dem Format zu extrahieren:

```
select st_x(st_astext(position))AS LAT,st_y(st_astext(position))AS LON from
tdp_fcd_raw limit 100;
```

Um diese Abfrage verstehen zu können, sollen erst die Funktionen von PostgreSQL erklärt werden.

- **st_x** Die Funktion gibt die X-Koordinate eines Punktes oder NULL, wenn der Punkt nicht abrufbar ist, zurück.
- **st_y** Diese Funktion gibt die Y-Koordinate eines Punktes oder NULL, wenn der Punkt nicht abrufbar ist, zurück.
- **st_astext** Diese Funktion gibt die WKT-Beschreibung[16] der Geometrie zurück**.**

Die anderen Schlüsselworte gehören zur SQL-Syntax:

- **select from** (Select) Daten aus (from) einer Tabelle abfragen.
- **as** Der Alias-Befehl hilft bei der Organisation der Ausgabe der Daten, in dem ein Stellvertreter vergeben wird, der anstelle der Spalte oder auch Tabelle zur Referenzierung genutzt werden kann.
- **limit** Dieses Schlüsselwort ermöglicht die Begrenzung der Anzahl der ausgegeben Zeilen auf den darauf folgenden Zahlenwert.

Zusammengeführt ergibt sich aus den Einzelelementen eine Anfrage von einer X- und einer Y-Koordinate in WKT-Format an die Spalte „position“, wobei X als LAT und Y als LON von der Tabelle „tdp_fcd_raw“ auf 100 Zeilen limitiert ausgegeben werden soll.

Der „status“ bezieht sich auf den mitgelieferten Status der FCD-Meldung, siehe Tabelle 4 „FCD Status“. Bei „speed“, sofern vorhanden, ist die aufgenommene Geschwindigkeit

[15] PostGIS ist eine Erweiterung für PostgreSQL, die die Möglichkeit bietet geografische Objekte und Funktionen zu nutzen. [10]

[16] Well-Known Text (WKT) ist eine Menschen lesbare Darstellung in diesem Fall von einer Geometrie die im Geodatenbanksystem nach ISO 19125 Standard beschrieben ist [78], z.B. ein Punkt (10,10) wird durch zwei Zahlen X und Y beschrieben.

abzulesen und „angle“ gibt den Winkel an, in dem sich das Fahrzeug relativ zum magnetischen Norden der aktuellen Position befindet.

Einer der Lieferanten dieser Daten ist Austrosoft[17], der die Daten im XML-Format[18] an den „DataImporter“ liefert. Die Gesellschaft für offene Systeme mbH (GefoS) ist ein weiterer Lieferant, der als Übertragungsformat entweder JSON[19] oder direkt REST nutzt. Der Representational State Transfer (REST) stellt hierbei die eleganteste Art der Datenübertragung an den „DataImporter“ dar. Mittels REST wird über eine eineindeutige Adresse eine spezielle Ressource genau identifiziert. Der Uniform Resources Identifier (URI) zeigt somit auf ein Angebot, das durch den Datenimport als Webservice bereitgestellt wird. Der FC-Datensatz wird dann mit entsprechenden Übergabeparametern an diese Adresse geschickt und in der stadtspezifischen Datenbanktabelle abgelegt. Der dritte Datenlieferant ist Microtek, der seine FC-Daten im File Transfer Protokoll (FTP) bereitstellt. Für die Akquirierung dieser Daten wurde ein kleines Script geschrieben, welches die Daten holt, verarbeitet und in die Tabelle „tdp_fcd_raw“ der jeweiligen Stadt schreibt. Die Daten können jetzt von der „FCD Process Unit“ weiterverarbeitet werden.

4.1.2. FCD Process Unit

Die „FCD Process Unit“ stellt innerhalb der FCD-Kette ein eigenes Modul dar, das wiederum in kleinere Module aufgeteilt ist. Die Komponenten sollen im folgenden Abschnitt in ihrem Aufbau, ihrer Arbeitsweise und Zusammenarbeit beschrieben werden.

Zuerst müssen die Daten für den „Trajektorizer“ aus der Datenbank aufgearbeitet werden. Die Bereitstellung für den „Trajektorizer“ stellt der „Filter“ sicher, der die Daten in geeigneter Form an das Modul liefert. Der „Trajektorizer“ arbeitet mit dem „Matcher“ zusammen, welcher die Daten mit dem Kartennetz verknüpf und somit einen Bezug zur realen Welt herstellt.

[17] Austrosoft ist eine Firma, die Flottenmanagement-Systeme anbietet. Die Firma wurde 1982 von Michael Weiss und Gerhard Tillich gegründet. [3]
[18] Die Extensible Markup Language (XML) ist ein durch das World Wide Web Consortium (W3C), in der XML-Spezifikation herausgegebener Standard.
[19] Die JavaScript Objekt Notation (JSON) ist ebenfalls eine Textform wie XML, nur komprimierter. Dabei ist jedes JSON-Dokument in gültigem JavaScript geschrieben und kann somit validiert werden (Eval Programmfunktion).

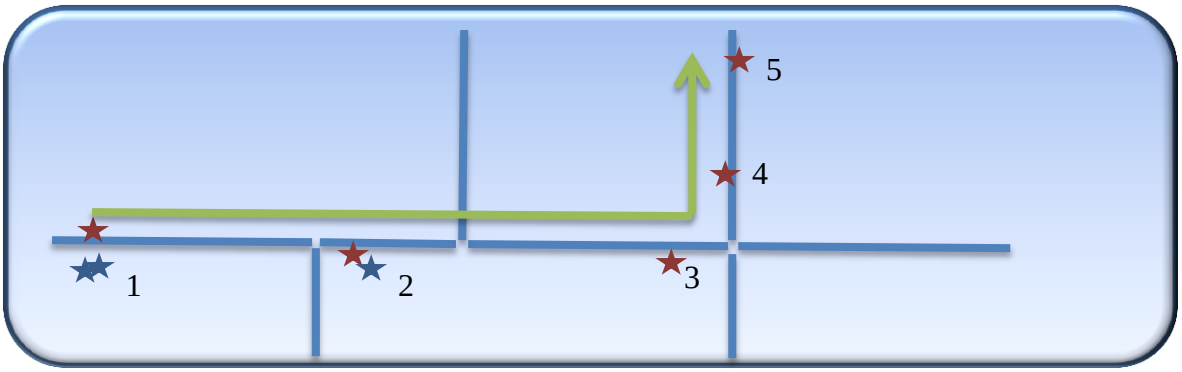


Abbildung 20: Trajektorizer

Die Abbildung 20 „Trajektorizer“ zeigt die Arbeitsweise des Moduls. Die Sterne innerhalb der Abbildung stellen FC-Daten dar, also Positionen zum Zeitpunkt t, die von einem Taxi erzeugt wurden. Die blauen Linien sind Straßen bzw. Kanten, die zum Straßennetz gehören. Der „Trajektorizer“ analysiert die FC-Daten und bringt diese in eine logische zeitliche Reihenfolge, da mehrere FC-Daten für denselben Punkt vorliegen können (siehe Abbildung 20 Punkt 1), wenn das Fahrzeug zum Beispiel an einer Ampel oder im Stau stand. Wenn dem so ist, entscheidet das Modul selbst, welche FC-Daten für den jeweiligen Punkt verwendet werden, in Abbildung 20 als rote Sterne dargestellt. Die verwendeten FC-Daten, die zu einer Trajektorie gehören, repräsentieren einen logischen Verbund, der grüne Pfeil.

Mit Hilfe des „Matchers“ werden die Positionen einer Trajektorie auf die Kanten (Edges) eines digitalen Straßennetzes projiziert, wie die folgende Abbildung zeigt:

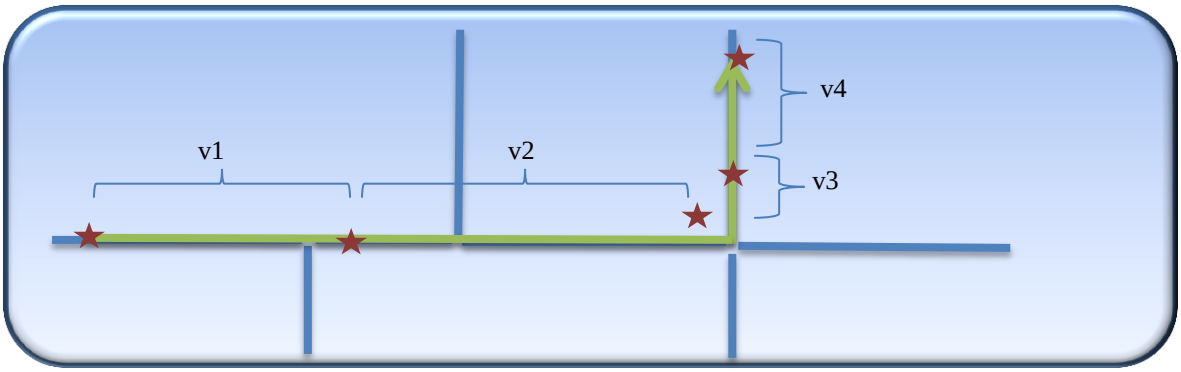


Abbildung 21: Matcher

Der „Matcher“ verwendet nur die FC-Daten der Trajektorie (rote Sterne), wie in Abbildung 21 zu sehen ist. Des Weiteren werden fehlende Kanten, die keine Positionsdaten besitzen, ergänzt. Nach dem Matching ist bekannt, welche Strecke das Taxi gefahren ist. Die erstellte Trajektorie (grüner Pfeil) mit den vom „Matcher“ erstellten Informationen wird an den „LinkSpeedGenerator“ übergeben, der die gefahrene Geschwindigkeit auf den Abschnitten

zwischen den FC-Daten bestimmt (v1 bis v4). Wie die Abbildung zeigt, kann eine Trajektorie in einer Kante beginnen, wenn z.B. das Taxi auf einem Parkplatz hält, der entsprechend in eine Straße mündet. Der „LinkSpeedGenerator“ erstellt nun auf Basis dieser Informationen die folgenden Tabellen bzw. Dateien. Dabei bezieht sich die Auswahl auf die in Abbildung 18 gezeigten Ausgabeformate.

KML-Datei

Die Keyhole Markup Language (KML) Datei ist eine Möglichkeit, Geodaten für den Client, zum Beispiel Google Earth und Google Maps zu beschreiben. Dabei bildet dieses Format mehrere geometrische Elemente ab. Damit ist es möglich, die bei FCD genutzten Linien, Punkte und Gebiete, in 2D und 3D zu modulieren. Dieses Format wird vom „LinkSpeedGenerator“ geliefert und wird zu Fehleranalysezwecken (Debugging) genutzt.

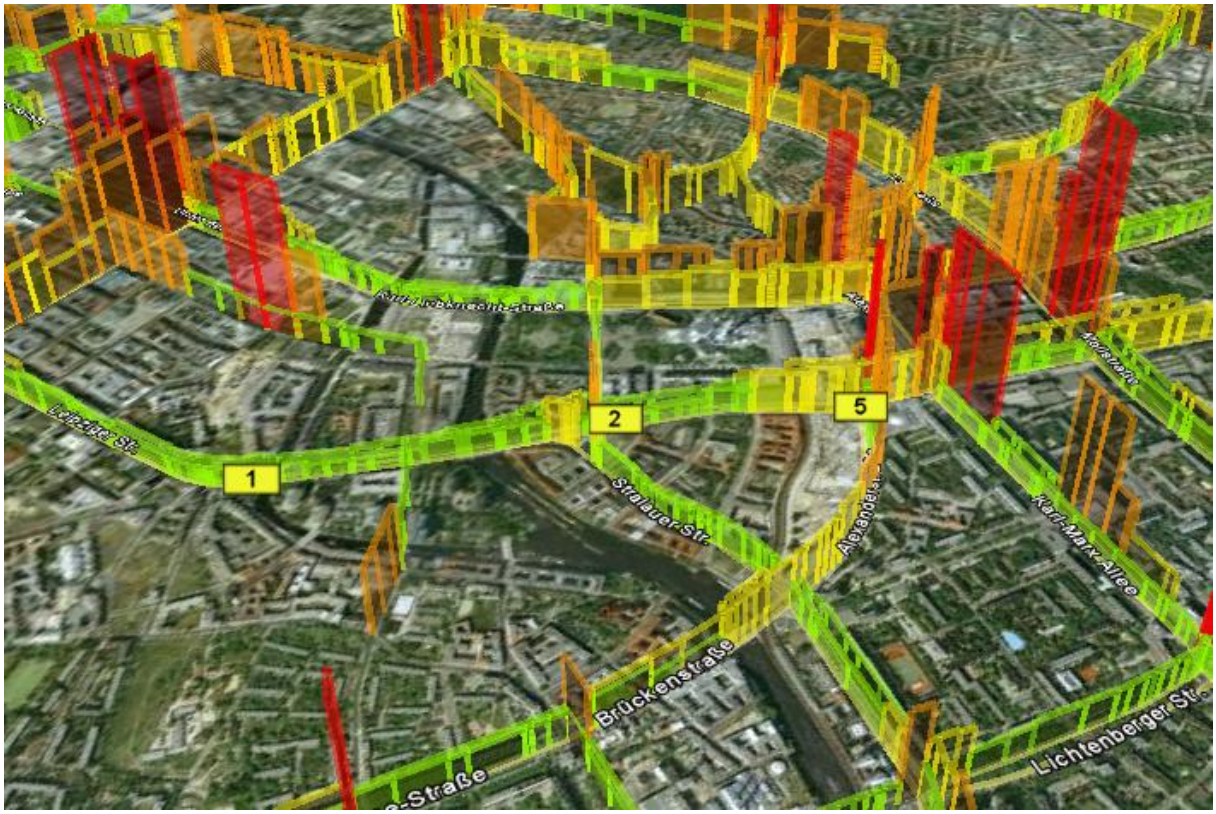


Abbildung 22: KML Beispiel

Die Abbildung 22 „KML Beispiel“ zeigt, wie die geometrischen Elemente beim Client dargestellt werden können. Die hier modulierte Verkehrslage zeigt, welches Potential von KML hinsichtlich seiner 3D Modellierungsfähigkeit zu erwarten ist. Wie zu sehen ist, können den Arealen unterschiedliche Texturen zugeteilt werden: Rot, grün, gelb, orange. Aber auch

eigene Bilder in PNG[20] oder JPG[21] können über die erstellten Flächen gelegt werden. Ferner besteht die Möglichkeit Animationen zu erstellen. In diesem Beispiel werden 4 Farben angezeigt, ein sogenannter vierstufiger Level Of Service (LOS). Die aktuelle Verkehrslage nutzt in der Shape-Datei lediglich 3 Farben zur Kennzeichnung der Verkehrssituation, ein dreistufiger LOS. Aus diesem Grund soll die Erklärung der Farbwahl und deren Zusammenhang mit der Verkehrssituation im folgenden Abschnitt beschrieben werden.

Shape-Datei

Das vom „LinkSpeedGenerator“ erstellte Shapefile dient ebenfalls der Modellierung von Geodaten. Dieses von ESRI[22] erstellte Format hat sich weitgehend durchgesetzt und gilt in vielen Fällen als Standard. [40]

Abbildung 23: Shape Beispiel

Die Abbildung 23 „Shape Beispiel“ zeigt die aktuelle Verkehrslage von Berlin (12:01 Uhr 28.06.2011). Das Shapefile beschreibt intern verschiedene geometrische Formen, die von verschiedenen Tools angezeigt werden können, in diesem Beispiel von Openlayers. Nach der Einbindung des Shapefiles wird dieses über dem eigentlichen Kartenmaterial als Layer angezeigt. Die grünen, gelben und roten Linien auf den Straßen repräsentieren die vom „LinkSpeedGenerator“ erstellte Übersicht der Verkehrslage wie folgt:

[20] Portable Network Graphics (PNG) ist ein Grafikformat für Rastergrafiken.
[21] Norm ISO/IEC 10918-1 (JPEG) [6]
[22] Environmental Systems Research Institute (ESRI) ist ein US-amerikanischer Softwarehersteller. [4]

Wenn die Differenz der erlaubten Geschwindigkeit zur Momentangeschwindigkeit eines FCD-Fahrzeuges in einer Trajektorie größer als die Hälfte ist, also mehr als 50 Prozent, wird diese Kante grün gefärbt, unter der Hälfte der erlaubten Geschwindigkeit und über einem Viertel gelb und bei einer Momentangeschwindigkeit unter 25 Prozent wird Rot den Abschnitt kennzeichnen. So wird bei einem Streckenabschnitt, auf dem 50 km/h erlaubt sind, aber nur 10 km/h gefahren werden, der Wert x = 20 Prozent wie folgt zu bestimmen sein:

$$\frac{100}{x} = \frac{50\ \mathrm{km/h}}{10\ \mathrm{km/h}}$$

Das Shape-File enthält die aktuellen Geschwindigkeiten, die an den Mapserver[23] übergeben werden. Mit dessen Hilfe bestimmt eine Map-Datei des Servers die entsprechenden Prozente nach der folgenden Formel:

$$\mathrm{x} = \frac{\text{Momentangeschwindigkeit km/h}}{\text{erlaubte Geschwindigkeit km/h}} * 100$$

Zusammengefasst ergibt sich aus diesem Abschnitt die Tabelle 5 „Färbungstabelle“ für alle Shapefiles, die dem Mapserver übergeben werden.

Tabelle 5: Färbungstabelle

Färbung der Kante	$\mathbf{x = \frac{Momentangeschwindigkeit\ km/h}{erlaubte\ Geschwindigkeit\ km/h} * 100}$
Grün	x > 50%
Gelb	50% < x < 25%
Rot	x < 25%

Diese Unterteilung richtet sich nach der Pflichtlektüre für Mitarbeiter im Institut für Verkehrssystemtechnik. [8] Diese Standardwerk mit dem Titel „Verkehr“ beschreibt die einzelnen Qualitätsstufen, den sogenannten „Level of Service“, der nach der Verkehrsdichte definiert wird. Die Literatur [8] selbst beschreibt sechs Unterteilungen im LOS, wobei immer zwei in der Färbungstabelle zusammengefasst werden. Demnach beschreibt grün innerhalb

[23] Mapserver ist ein Server der spezialisiert ist Services anzubieten, die Geodaten verarbeiten und bereitstellen können.

der Übersicht eine gute Qualität der Verkehrsdichte, also freie Fahrt, gelb zähfließenden Verkehr und rot Stau. Den zähfließenden Verkehr kennzeichnet eine hohe Belastung des Ablaufes im Verkehr, dabei ist die Verkehrslage noch stabil, also ein Vorrankommen noch möglich. Die erhöhte Interaktion der Fahrzeuge miteinander führt jedoch zu einer starken Reduzierung des zur Verfügung stehenden Freiraumes eines jeden Teilnehmers. [8][S107] Das letzte Szenario beschreibt eine Verkehrsdichte, in der die Steuer- und Regelungsanlagen nicht mehr fähig sind, die vorhandenen Kapazitäten zu verwalten, was zu einem Ausfall der Mobilität führt, dem Stau. [8][S108]

Datenbanktabelle-Matches

Die Tabelle „tdp_fcd_matches“ wird ebenfalls vom „LinkSpeedGenerator“ gefüllt. Die Abbildung 24 zeigt einen Ausschnitt aus dieser Tabelle für die Stadt Berlin.

Daten editieren - PostgreSQL 8.4 (129.247.218.180:5432) tdp_berlin - navteq_2011_q2.tdp_fcd_matches

Datei Bearbeiten Anzeigen Werkzeuge Hilfe

100 Zeilen.

	traj_id bigint	local_time [PK] timestar	veh_id [PK] integer	edge_id bigint	distance_from_eb smallint	local_speed smallint	travel_speed smallint	travel_dist smallint	quality smallint
1	1306937928670	2011-06-06	129243	51086525	17	255	29	6	70
2	1306937928678	2011-06-06	244673	844746513	0	255	19	64	70
3	1306937928606	2011-06-06	772214	-51086242	0	255	6	101	70

Abbildung 24: Tabelle „tdp_fcd_matches“

Die erste Spalte „traj_id“ beinhaltet die Trajektorie-ID, die vom „Trajektorizer“ vergeben wurde. Spalte „local_time“ enthält den zugehörigen Zeitstempel. Die Tour-ID ist in der Spalte „veh_id“ zu finden. Die Spalte „edge_id“ gibt zu dem Match gehörende Kanten im digitalen Straßennetz an, „local_speed“ kennzeichnet die von GPS gemessene Geschwindigkeit, wobei 255 einen Dummywert[24] darstellt. Der „travel_speed“ repräsentiert die durchschnittliche Reisegeschwindigkeit des Fahrzeuges auf der Kante und „quality“ die Qualität der Messung. Die Spalte „distance_from_eb“ gibt die Strecke in Metern an, die der Matchingpunkt vom Kantenanfang entfernt ist, „travel_dist“ gibt die Entfernung vom Anfang der Trajektorie an. Also würden die Werte 17 Meter (Spalte 4) und 6 Meter (Spalte 7) für Zeile 1 wie folgt zu interpretieren sein.

[24] Dummywert ist eine Größe die nicht gemessen wurde (fiktiv) und lediglich als ein Platzhalter zu verstehen ist.

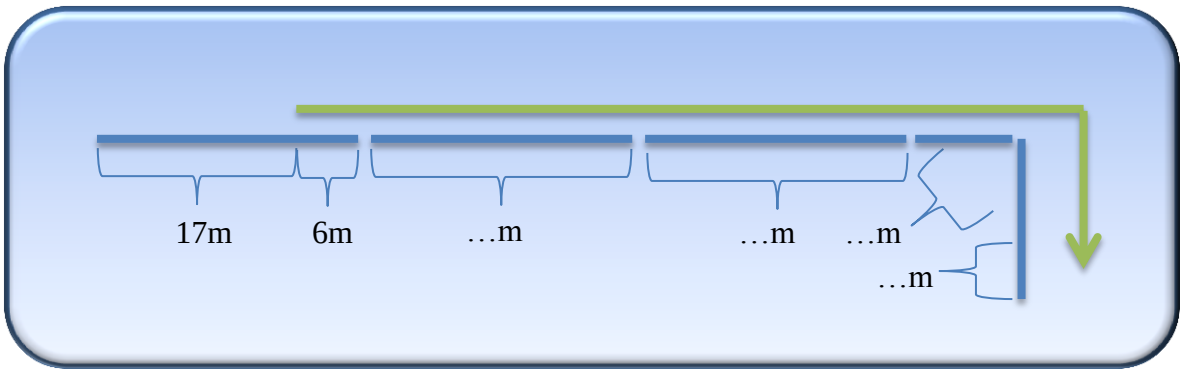


Abbildung 25: Matches Tabelle Beispiel

Die Kanten (blaue Linien) wurden von der Tour 129243 aus Zeile 1, wie in Abbildung 25 „Matches Tabelle Beispiel“ zu sehen, abgefahren (grüner Pfeil). Das Fahrzeug fuhr 17 Meter nach dem Beginn der Kante (erste blaue Linie) auf dem Abschnitt los, weil z.B. ein Fahrgast eingestiegen ist und verließ 6 Meter vom Anfang der Trajektorie entfernt (grüner Pfeil) die Kante wieder, da diese dort endet.

Datenbanktabelle-On_Edges

Bei der Tabelle „tdp_fcd_on_edges“ kommen die Spalten „cover“, „minute_of_week“, „distance_to_ee“ und „no_trafficstate“ hinzu, die anderen Spalten sind gleichbedeutend mit der Matches Tabelle, wie die Abbildung 26 zeigt.

Daten editieren - PostgreSQL 8.4 (129.247.218.180:5432) tdp_berlin - navteq_2011_q2.tdp_fcd_on_edges

Datei Bearbeiten Anzeigen Werkzeuge Hilfe

100 Zeilen.

	traj_id bigint	local_time [PK] timestar	veh_id [PK] integer	edge_id [PK] bigint	travel_speed smallint	cover smallint	minute_of_week smallint	distance_from_eb smallint	distance_to_ee smallint	quality smallint	no_trafficstate boolean
1	1306937928670	2011-06-06	129243	51086472	29	43	0	0	111	100	FALSE
2	1306937928670	2011-06-06	129243	51086525	29	26	0	17	0	70	FALSE
3	1306937928678	2011-06-06	244673	844746513	19	51	0	0	64	70	FALSE

Abbildung 26: Tabelle „tdp_fcd_on_edges“

Der „cover“ Wert drückt den Abdeckungsgrad der Kante in Prozent aus. Bei einem Wert von 100 wurde die gesamte Kante befahren. Bei dem Wert 26 aus Zeile 2 wurden nur 26 Prozent der Kante abgefahren, die Abbildung 27 zeigt das Verhältnis.

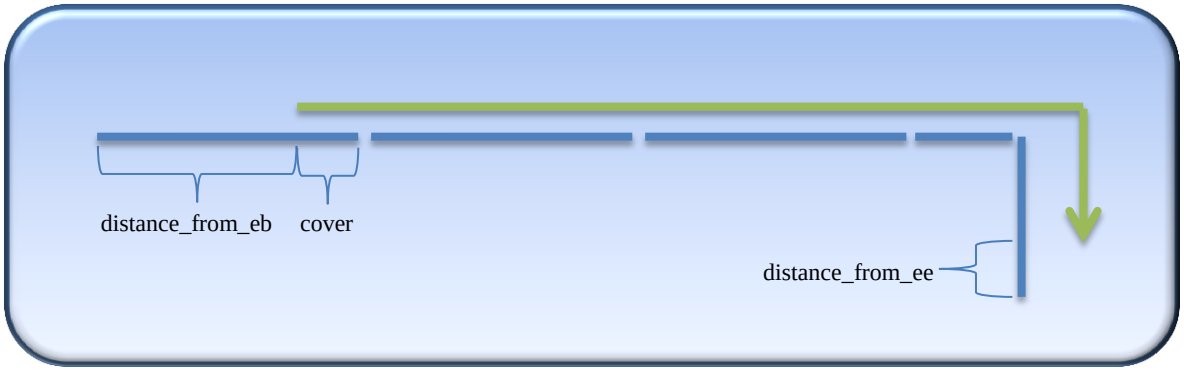


Abbildung 27: Edges Tabelle Beispiel

Der Bezugswert der Daten ist in dieser Tabelle immer die Kante. Für jede Kante gibt es somit eine Reisegeschwindigkeit („travel_speed“), die von der entsprechenden Tour (veh_id) erzeugt wurde. Diese Kante gehört zu der Trajektorie („traj_id“), die, siehe z.B. Zeile 1, von Anfang an befahren wurde („distance_from_eb“ = 0) und 111 Meter vor Ende verlassen wurde („distance_from_ee“ = 111). Die Spalte „minute_of_week“ beschreibt die Minute der Woche. Da ein Tag 1440 Minuten besitzt, ist der Wert für 7 Tage, also eine Woche = 10080 Minuten. Der Counter wird beim Überschreiten dieses Wertes wieder auf 0 gesetzt und eine neue Woche wird in Minuten gezählt. Der Vorteil in der Bestimmung dieser Größe liegt bei der schnellen Abfrage für gewisse Wochentage aus der Tabelle. So würde man alle Montage mit einer entsprechenden „where“-Bedingung[25] erhalten, bei dem der Wert von „minute_of_week“ zwischen 0 und 1440 liegt. Die letzte Spalte „no_trafficstate“ ist ein Flag[26], das angibt, ob der Wert nicht mit in die Bestimmung der Verkehrslage eingeflossen ist. Bei „FALSE“, wie in der Abbildung, würde dies bedeuten, dass die Zeile mit in die Berechnung eingeflossen ist.

Datenbanktabelle-Current_Speeds

Daten editieren - PostgreSQL 8.4 (129.247.218.180:5432) - tdp_berlin - navteq_2011_q2.tdp_fcd_current_speeds

Datei Bearbeiten Anzeigen Werkzeuge Hilfe

100 Zeilen.

	edge_id [PK] bigint	last_update timestamp without time zone	travel_speed smallint	veh_count smallint	quality smallint
1	-850288241	2011-06-21 20:11:10	32	1	1
2	-850288240	2011-06-21 20:11:10	32	1	1
3	-850285629	2011-06-29 09:17:11	25	2	1

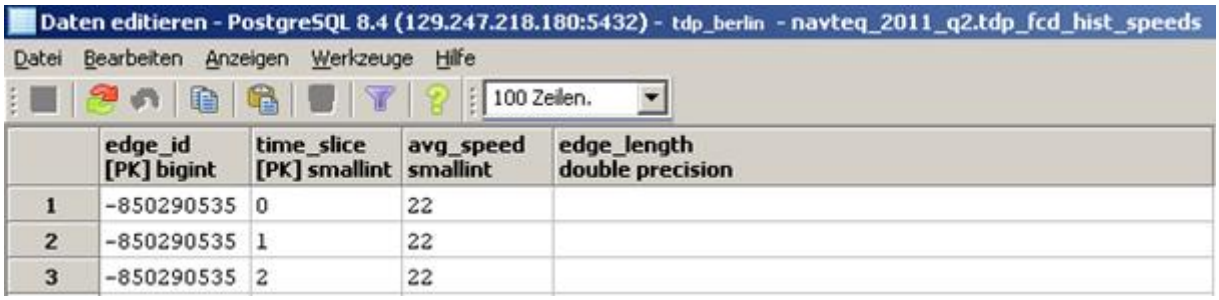
Abbildung 28: Tabelle „tdp_fcd_ current_speeds"

Die Abbildung 28 „tdp_fcd_current_speeds“ repräsentiert nur die aktuelle Verkehrslage der jeweiligen Region. Hier wird jeder Kante eine durchschnittliche Reisegeschwindigkeit zugeordnet, die auf Basis der in „veh_count“ gezählten Fahrzeuge bestimmt wird. So würde, z.B. „travel_speed“ = 32km/h entstehen, wenn entweder ein Fahrzeug mit 32km/h die Kante befährt oder 2 Fahrzeuge mit jeweils 30km/h und 34km/h diese befahren. Dies würde unter der Voraussetzung geschehen, dass alle Werte das gleiche Alter haben, sonst würde abhängig vom Alter des Datensatzes, eine Gewichtung die Geschwindigkeit zu Gunsten der jüngeren Daten verschieben. Die Zeile 3 mit „travel_speed“ = 25 km/h zeigt so einen erstellten Wert,

[25] Where-Bedingung entspricht dem aus der SQL-Welt entsprechenden Verfahren, um an ein gestelltes Select (Auswahl) weitere KO-Kreterin zu knüpfen.
[26] Flag entspricht in der Informatik einem Datentyp der nur die Werte 0 oder 1 annehmen kann, also wahr (true = 1) oder falsch (false = 0).

der aus der Geschwindigkeit zweier Fahrzeuge bestimmt wurde. Ferner ist in der Spalte „last_update“ zu lesen, wann das letzte Fahrzeug die Kante befahren hat.

Datenbanktabelle-Hist_Speeds



	edge_id [PK] bigint	time_slice [PK] smallint	avg_speed smallint	edge_length double precision
1	-850290535	0	22	
2	-850290535	1	22	
3	-850290535	2	22	

Abbildung 29: Tabelle „tdp_fcd_hist_speeds"

Die Abbildung 29 zeigt die Tabelle „tdp_fcd_hist_speeds“ welche ähnlich der „tdp_fcd_current_speeds“ Tabelle ist und deren Werte für einen definierten Zeitraum, z.B. vom 01.05.2011 bis zum 31.10.2011, aus der Tabelle „tdp_fcd_on_edges“ aggregiert werden. Die Spalte „time_slice“ gibt an, wie der entsprechende Zeitraum aufgeteilt ist. Der Wert ist aus der Tabelle „fcd_config“ abzulesen und standardmäßig 60. Das bedeutet, dass in 60 Minutenabschnitten gezählt wird, übertragen auf die Abbildung sind die ersten drei Stunden der Woche aufgezeigt. Die Durchschnittsgeschwindigkeit für jede Kante in der jeweiligen Stunde ist in der Spalte „avg_speed“ abzulesen. Diese Tabelle enthält die Zusammenfassung aller Daten für den gewählten Zeitraum, für einen typischen Wochentag, der je nach Aufteilung des Wertes aus der „fcd_config“ Tabelle zusammengefasst wird. Ein Beispiel soll das Verfahren verdeutlichen.

Angenommen, es soll der Zeitraum vom 01.05.2011 bis 31.10.2011 als repräsentativer Ausschnitt für den Sommer 2011 gewählt werden, um daraus ein typisches Verkehrsaufkommen für eine bestimmte Route zu bestimmen. Der Zeitraum besitzt 26 Montage, diese werden zusammengefasst und zu einem Montag aggregiert. Dieser Montag hat 24 Datensätze in der Tabelle, die ihn repräsentieren. Da die Zeitaufteilung nach Tabelle „fcd_config“ 60 ist und bei einer Unterteilung eines Tages in 60 Minuten bzw. in Stunden, 24 repräsentative Datensätze benötigt werden, um einen gesamten Tag abzubilden. Dieses Verfahren wird mit jedem Tag einer Woche wiederholt. So repräsentieren immer 24 Datensätze dieser Tabelle einen typischen Tag in diesem Zeitraum. Bei 7 Tagen mit je 24 Datensätzen sind insgesamt 168 Datensätze gespeichert. Bei der Bestimmung der typischen Verkehrslage für einen Dienstag 9 Uhr muss der Datensatz mit dem „time_slice“ von 32 aus der Datenbank gelesen werden, da in der Informatik von 0 gezählt wird. So gehören die

Datensätze 0 bis 23 zum Montag und 9 Datensätze weiter finden sich die gesuchten Werte für den Dienstag.

4.1.3. Service Provider

Der zuvor auf Basis der Abbildung 18 „FCD-Kette“ beschriebene Aufbau zeigt den Verlauf der FC-Daten vom Produzenten bis hin zur persistenten Speicherung. Das Modul zur Weitergabe an die Klienten lässt sich wiederum in Einzelbereiche aufgliedern, wie die folgende Abbildung zeigt.

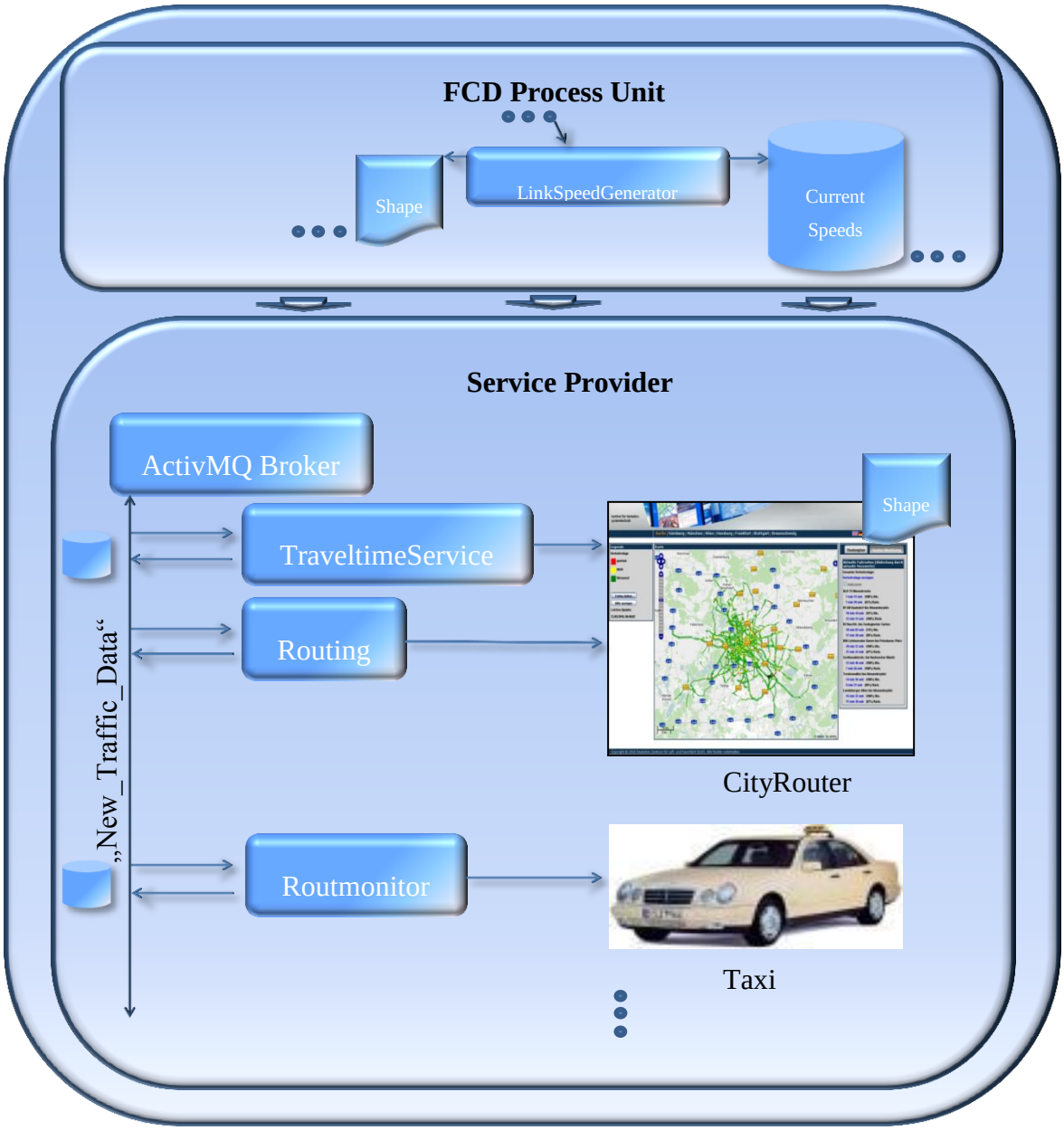


Abbildung 30: FCD Darstellung & Herausgabe der Daten

Die Abbildung 30 „FCD Darstellung & Herausgabe der Daten“ stellt die im Bereitstellungsmodul laufenden Services dar. Für die Kopplung der „FCD Process Unit“ mit den entsprechenden Diensten des Webservers wird die ActivMQ Architektur von Apache genutzt. Diese sieht eine sehr lose Verbindung der Teilnehmer vor. Der Message Broker[27] benutzt Messagequeues[28], die den asynchronen Nachrichtenaustausch sicherstellen. Asynchron deshalb, weil die „FCD Process Unit“ ständig neue Daten generiert und weder die „FCD Process Unit“ noch die Datenbank durch das Warten auf nachgestellte Services beinträchtig werden soll. Die Information, dass neue Daten erstellt wurden, wird dem „ActiveMQ Broker“ übergeben, der die Nachricht in einer Messagequeue speichert und anschließend per Broadcast[29] sendet. Auf Basis dieser Messagequeues werden intern Jobs gebildet, die nacheinander abgearbeitet werden. Die Nachricht „New_Traffic_Data“ beschreibt intern, dass neue FC-Daten vorliegen. Diese Information wird an die angemeldeten Services weitergegeben. Die Services „TraveltimeService“, „Routing“ und „Routmonitor“ werden von verschiedenen Clients genutzt, z.B. dem „CityRouter“ oder einem Routenüberwachungsservice, der Reisezeiten auf ausgewählten Routen direkt im Taxi anzeigt. Je nach Auslastung und verstreichender Zeit, fragen die Services bei der Datenbank die neuen Verkehrsdaten an, wenn sie per Broadcast zuvor informiert wurden, dass neue Nachrichten vorhanden sind. [41]

Der „TraffiltimeService“ kann genutzt werden, um Reisezeiten für einzelne Kanten oder ganze Route anzufragen. Der „Routingmonitor“ hingegen liefert Reisezeiten für überwachte Routen. Der „Cityrouter“ nutzt diese Services und stellt deren Funktionalität im Web Frondend dem Nutzer zur Verfügung. Die Darstellung der aktuellen Verkehrslage auf Basis von FC-Daten wird als Layer mittels Shapefile über die selbst erstellte Karte gelegt.

[27] Message Broker ist als ein Nachrichtenverteiler zu verstehen.
[28] Eine Messagequeue ist eine Nachrichtenreihe, die der Broker intern verwaltet.
[29] Broadcast beschreibt das Verfahren, bei dem eine Nachricht an alle Teilnehmer eines Computernetzwerkes versendet wird.

4.2. Sollzustand / Lösungskonzept

In diesem Abschnitt soll der theoretische Ansatz der TMC-Verarbeitungskette innerhalb des DLR aufgezeigt werden. Die tatsächliche praktische Umsetzung und deren detaillierte Beschreibung erfolgt in den folgenden Kapiteln.

Die unter Abschnitt 4.1. „ Istzustand“ beschriebene FCD-Kette muss mit der zu erstellenden TMC-Kette sinnvoll verknüpft werden.

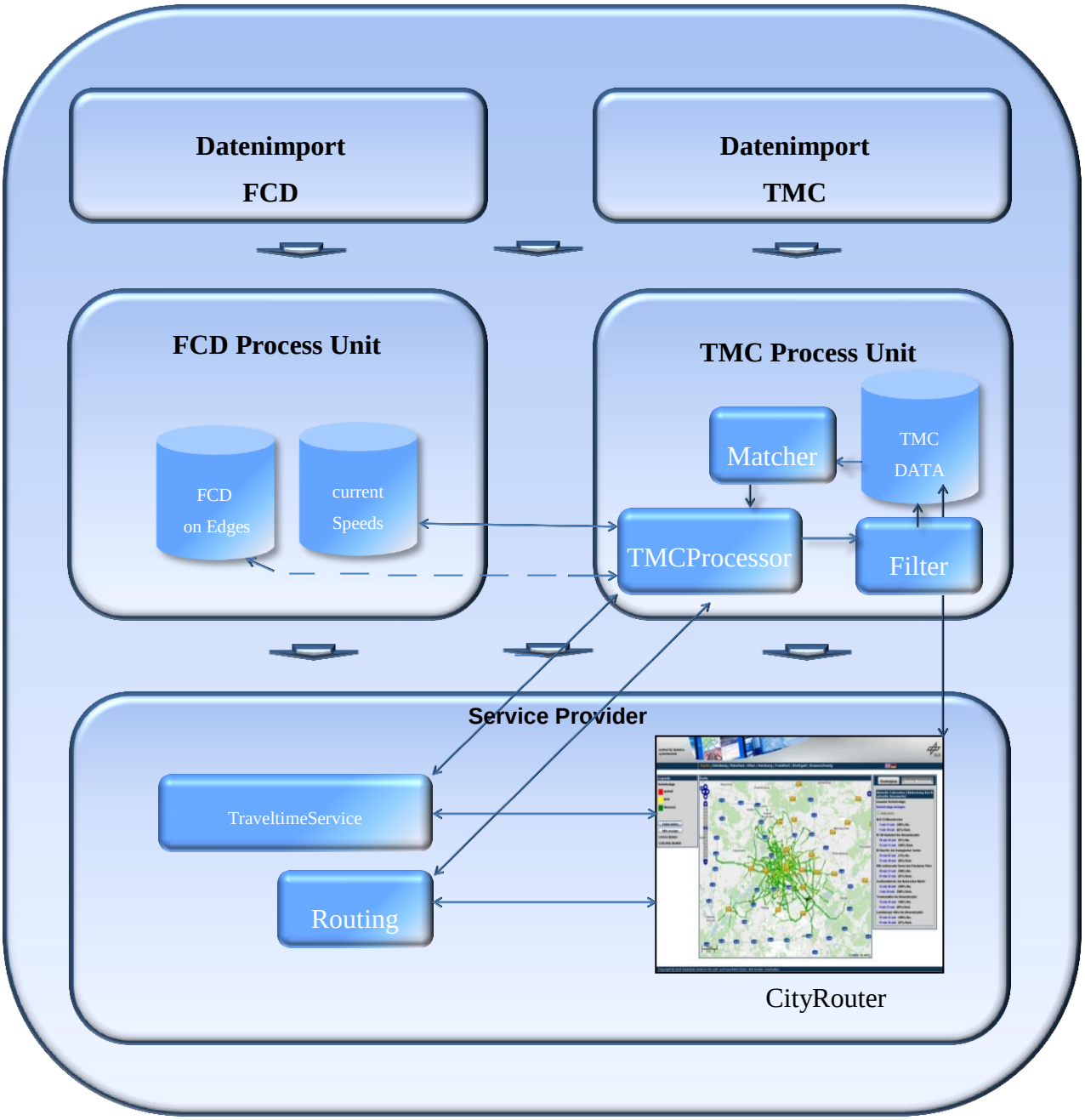


Abbildung 31: TMC & FCD

Abbildung 31 „TMC & FCD“ zeigt in welcher Form die Verknüpfung der FCD und der TMC-Kette sinnvoll ist. Da beide Systeme, wie gezeigt wurde, unterschiedliche Datenquellen besitzen, ist ein separater Datenimport geeignet. Die Verarbeitungseinheiten beider Systeme sollen hingegen auf Elemente der jeweils anderen zugreifen können. Die „TMC Process Unit“ hat ein Verarbeitungsmodul, in der Abbildung 31 als „TMCProcessor“ gekennzeichnet, das entweder die Tabellen oder die Service nutzt, um TMC-Daten mit FCD zu validieren. Dieses Modul wird die durch den „Datenimport“ bereitgestellten Informationen gefiltert in eine Datenbank schreiben. Dies ist sinnvoll, um die gezeigten Zusatzinformationen ebenfalls zu speichern und so eine für den Menschen besser lesbare Struktur in den Tabellen zu schaffen. Die Module werden dabei, ganz ähnlich wie in der „FCD Process Unit“, in Wechselwirkung mit der entsprechenden persistenten Speicherung stehen. Des Weiteren wird der „TMCProcessor“ nicht in Wechselwirkung mit einem Verarbeitungsmodul, wie dem „Trajektorizer“ oder dem „LinkSpeedGenerator“ stehen, da diese Module um die entsprechenden Schnittstellen erweitert werden müssten. Die Erweiterung von Software um neue Funktionen ist schwierig. Oft muss das laufende System unterbrochen werden, um die Erweiterungen zu testen. Ferner führen etwaige Fehler, wie ein unterlassenes Exceptionhandling[30], zu Ausfällen einer Verarbeitungskette, die zuvor einwandfrei funktionierte. Weiterhin ist zu bedenken, dass fremde Programme eine größere Einarbeitungszeit benötigen und bei einer mangelnden Dokumentation Kleinigkeiten übersehen werden können, die in Folge dessen zu Fehlern führen. Sinnvoll ist es, direkt das Endprodukt der Module zu nutzen, also die erstellten Dateien oder Tabellen.

Die Tabelle „tdp_fcd_current_speeds“ repräsentiert die aktuelle Verkehrslage der jeweiligen Stadt und bietet sich somit als Abgleich mit den TMC-Daten an. Die in der Tabelle angegebenen Kanten enthalten Geschwindigkeiten, die durch geeignetes Matching auf die TMC-Location zu einem Abgleich genutzt werden können. Wenn eine TMC-Staumeldung durch den Datenimport eingeht, muss der Location-Code auf die entsprechende Kante übertragen werden, dies kann ein „Matcher“ bereitstellen, genaueres in Kapitel 7. Bei der Analyse werden relevante Daten, die noch zu spezifizieren sind, verglichen, um zu ermitteln, ob die TMC-Meldung stimmt bzw. ob die FC-Daten einer weiteren Überprüfung bedürfen. Darüber hinaus ist es auch möglich, direkt den „TravelTimeService“ vom „Service Provider“ zu nutzen, der eine Reisezeit für eine Route erstellen kann. Auch hier muss die TMC-Meldung mit einer Kante verknüpft werden, um anschließend diese an den Service als Route

[30] Der Begriff Exceptionhandling meint hier die Ausnahmebehandlung, die den Umgang mit unerwarteten auftretenden Zuständen in Programmen beschreibt.

zu übergeben. Die erhaltene Reisezeit bezieht sich indirekt ebenfalls auf die Tabelle „tdp_fcd_current_speeds“, die zu ihrer Bestimmung herangezogen wird. Der Service gibt dabei nicht nur die aktuelle Reisezeit an, sondern auch die schnellste Zeit, die zu erwarten ist, wenn keine Verkehrsstörung innerhalb dieser Route im FCD-System vorliegt.

Die Nutzung der anderen Tabellen bzw. Dateien oder Services erscheint bei der ersten Betrachtung nicht sinnvoll. Die Dateien beinhalten zwar die Verkehrslage, aber die Formate benötigen einer weiteren API zum Auslesen bzw. Parsen der enthaltenen Informationen. Die Tabellen hingegen können direkt über die JDBC-API[31] ausgelesen werden, da diese Programmierschnittstelle vom „TMCProcessor“ auf jeden Fall genutzt werden muss, um die TMC-Meldungen zu speichern. Die Herausforderung ist, dass die zu erwartenden relevanten Informationen über Geschwindigkeiten nicht in allen Tabellen enthalten sind, wie bei „tdp_fcd_matches“ und „tdp_hist_speeds“, oder zu viele nicht relevante Daten enthalten sein können, wie in der Tabelle „tdp_fcd_on_edges“. Wobei die zuletzt genannte Tabelle bei einer noch nicht fest definierten Generierung der neuen Daten von der weiteren Betrachtung und somit einer Nutzung dieser nicht ausgeschlossen werden kann, da augenscheinlich nicht relevante Daten in diesem Zusammenhang ihren Anspruch auf Relevanz noch gerecht werden könnten. Die entsprechende Linie wird aus diesem Grund in der Abbildung unterbrochen dargestellt, um weitere Möglichkeiten noch berücksichtigen zu können. Ein genaues Bild über die entworfene Kette und deren Interaktion mit FCD zeigt Kapitel 7. „Generierung einer neuen Datenbasis“. Sicher ist, dass die Darstellung der neuen Daten sich an dem bestehenden System bedienen und die TMC-Daten in dem Web Frontend des CityRouters anzeigen wird.

[31] Java Database Connectivity application programming interface (JDBC-API) ist eine Programmierschnittstelle für Java, die es dem Anwender ermöglicht auf Datenbanken zuzugreifen.

4.3. Verwendete Architekturen

Die erstellten Module der FCD-Kette sind in Java geschrieben. Trotzdem soll bei der Auswahl einer geeigneten Programmiersprache nicht vorschnell gehandelt werden, um eine möglichst hohe Effizienz der Bewältigung der bevorstehenden Aufgabe zu gewährleisten. Die Kriterien für die Messgrößen zur Bestimmung der geeignetsten Programmiersprache sind: Übersetzung, objektorientierte Programmierung, Abstraktion, Syntax, Standardbibliotheken und eigenes Wissen und Erfahrungen. Da lediglich Erfahrungen in C und dessen Erweiterung C++, C# sowie Java und Python vorliegen, können andere Sprachen in die Betrachtung nicht mit aufgenommen werden, da jedes der Kriterien zu erfüllen ist.

Tabelle 6: Vergleich von Programmiersprachen [42]

Programmier-sprache	**Übersetzung**	**OOP**	**Abstraktion**	**Syntax**	**Standard-bibliothek**	**Erfahrung in Jahren**
C	Compiler	nein	maschinennah (u.A. durch Inlineassembler, kein OOP); kein Garbage Collector	wenige Schlüsselwörter, viele Symbole; teilweise sehr abstrakt	sehr minimal	2
C++	Compiler	ja	sehr unterschiedlich: sowohl maschinennah (Inlineassembler möglich), als auch sehr abstrakt (OOP und generische Programmierung); kein Garbage Collector	C-basierend	generisch; mittel	3
C#	Compiler/ Bytecode-Interpreter	ja	abstrakt; bietet im unsafe-Modus allerdings Systemfunktionen	C-basierend	umfangreich	1
Java	Compiler/ Bytecode-Interpreter	ja	sehr abstrakt (keine Systemfunktionen etc.)	C-basierend	sehr umfangreich	7
Python	Interpreter	ja	abstrakt	Viele Schlüsselwörter, kaum Symbole;	sehr umfangreich	3

				Einrückungen tragen Semantik; ergonomisch		

Die Tabelle 6 „Vergleich von Programmiersprachen“ zeigt bis auf die letzte Spalte ein Abbild der unter der Quelle angegebenen Tabelle. [42] Ersichtlich wird, dass C keine objektorientierte Programmiersprache ist und mit seinem geringen Sprachumfang sowie mangelnden Standardbibliotheken keine gute Wahl darstellen würde. C++ entfällt wegen den im Vergleich doch wenigen Standardbibliotheken und der mittleren Erfahrung im Umgang mit dieser Sprache. C-Sharp lehnt sich an Java an und würde mit seinen doch umfangreichen Bibliotheken eine gute Alternative darstellen. Doch der Mangel an Erfahrungen mit dieser Sprache führt zu einem erheblichen Mehraufwand an Zeit bei der Programmierung. Die noch ausstehenden Sprachen sind Python und Java. Python ist eine Interpreter-Sprache. Das bedeutet, dass der Quellcode nicht in eine ausführbare Datei umgewandelt wird, sondern beim Ausführen immer direkt eingelesen, interpretiert und ausgeführt werden muss. Dies verringert die Ausführungsgeschwindigkeit der Programme im Vergleich zu neueren Verfahren, die bei Java eingesetzt werden. In Java übersetzt der Compiler den Quellcode in eine Class-Datei, die den sogenannten Bytecode enthält. Dieser Code ist ein maschinenlesbares HEX-Format, das von der Java Virtuellen Maschine (JVM) zur Laufzeit interpretiert und ausgeführt werden kann. Der Vorteil dieses Verfahrens ist nicht nur das performantere Verhalten der Anwendung, da der ByteCode für eine Maschine schneller zu verarbeiten ist, sondern auch dessen Plattformunabhängigkeit. Dies ist möglich, da die JVM eine Schnittstelle zu jedem System herstellt, auf dem sie selbst läuft. Bei dem Ausführen von ByteCode wird die Anwendung in einer eigenen virtuellen Maschine ausgeführt, die wiederum etwaige Steuerbefehle für das zugrundeliegende System übersetzt. Des Weiteren wurden in Java mehr als das doppelte an zeitlicher Erfahrung gesammelt, die weit über die Grundlagen der Programmierung hinausgehen. Sicher sind erlernte Entwurfsmuster auf andere Sprachen übertragbar, der Umgang mit der Sprache jedoch nicht. Aus den genannten Gründen bietet sich Java in dieser Arbeit als Werkzeug für die zu lösenden Aufgaben an.

4.4. Problemanalyse

Eine Problemanalyse soll zur Sensibilisierung des Themas beitragen und bewusst auf kritische Abschnitte hinweisen, um schwerwiegende Fehler zu vermeiden.

4.4.1. Datenimport

Das Empfangen der TMC-Meldungen stellt eine Herausforderung dar, da die Meldungen aber nicht die Empfänger im EN-ISO-14819 Standard definiert sind. Somit ist es möglich, dass ein TMC-Empfänger ein eigenes, nicht zugängliches Protokoll verwendet, um den RDS-Datenstrom zu interpretieren und zu decodieren. Dieser Problematik könnte man entgehen, indem ein entsprechender Empfänger selbst entworfen und gebaut wird. Der zeitliche Aufwand, um sich dieses Wissen anzueignen und umzusetzen, beträgt mehrere Wochen und ist im zeitlichen Rahmen dieser Arbeit nicht realisierbar. Eine andere Lösung ist die Wahl eines Empfängers, der ein Protokoll nutzt, dass bekannt ist oder zu dem eine Zugangsmöglichkeit besteht. Wobei der Erhalt einer möglichen API den optimalen Fall darstellen würde. Dies würde auch ein Abwenden von Java rechtfertigt, da eine Programmierschnittstelle, die das Ansprechen, Einstellen und Decodieren des Empfängers bzw. des Datenstroms ermöglicht, eine erhebliche Zeitersparnis mit sich bringt.

4.4.2. TMC Process Unit

Eine weitere Herausforderung ist die Interpretation der TMC-Daten. Es könnten, beispielsweise Zusatzinformationen, nicht richtig gedeutet oder in den entsprechenden Tabellen nicht eindeutig referenziert werden. Die größte Herausforderung ist die sinnvolle Verknüpfung der TMC-Daten mit den FCD-Daten. Die dazu zu bestimmenden Kriterien sollen dabei nicht nur theoretisch anwendbar, sondern auch praktisch messbar sein.

5. Empfangen der TMC-Meldungen

Im Folgenden soll beschrieben werden, wie TMC-Meldungen empfangen werden können und welche Herausforderungen es dabei gibt. Ferner wird auf die Informationenverarbeitung eingegangen. Weiter zu erklären ist, wie Informationen bereitgestellt werden können und welche Art der Speicherung sich aus welchen Gründen anbietet.

5.1. Datenimport TMC

Bei der Auswahl des richtigen TMC-Empfängers ist das wichtigste Kriterium ein Gerät zu wählen, dass einen vorinstallierten RDS-Decodierer besitzt. Dieser ermöglicht bereits bei dem Empfang des RDS-Datenstroms eine automatische Block-Synchronisation sowie eine Fehlererkennung und Korrektur. Diese Vereinfachung ist legitim, da der Fokus der Arbeit auf die Verarbeitung von TMC-Meldungen und dessen Integration in das DLR-System besteht und nicht auf dem Empfang von TMC-Meldungen selbst.

Abbildung 32: BT-465 UTE Navilock [43]

Eine kostengünstige Möglichkeit zeigt die Abbildung 32 von Navilock. Der „BT-465 UTE – Bluetooth/USB GPS/TMC“ Empfänger besitzt die geforderten Anforderungen: [43]

- USB Stick Ausführung mit Bluetooth Funktion
- Kein Akku, daher auch im Dauerbetrieb nutzbar
- MTK MT3318 - 32 Kanäle - Bluetooth GPS-Empfänger
- TMC basierend auf GNS 3.0 Protokoll
- NMEA-0183 Protokoll
- Abmessungen (Breite x Tiefe x Höhe) 6.1 cm x 2.7 cm x 0.9 cm
- Lieferumfang: UKW Wurfantenne/ Kurzstabantenne

Auf der Webseite von Navilock wird mit einem einfachen lesen der Daten geworben. Zitat: „Die Daten des Empfängers können ganz einfach über die serielle Schnittstelle Ihres Computers empfangen werden.“ [43] Die daraus resultierende Annahme, dass das Auslesen der seriellen Schnittstelle in Klartext möglich ist und einfach zu programmieren sei, zeigt einen Erkenntnisprozess, der der Annahme geschuldet ist, dass die entsprechenden Protokolle über RxTx[32]-API[33] zu lesen sind.

Beim Auslesen des Gerätes direkt an der USB-Schnittstelle mittels RxTx-API zeigten sich keine verwertbaren TMC-Daten. Dennoch konnte das GPS-Signal empfangen werden, wie die folgende Abbildung zeigt.

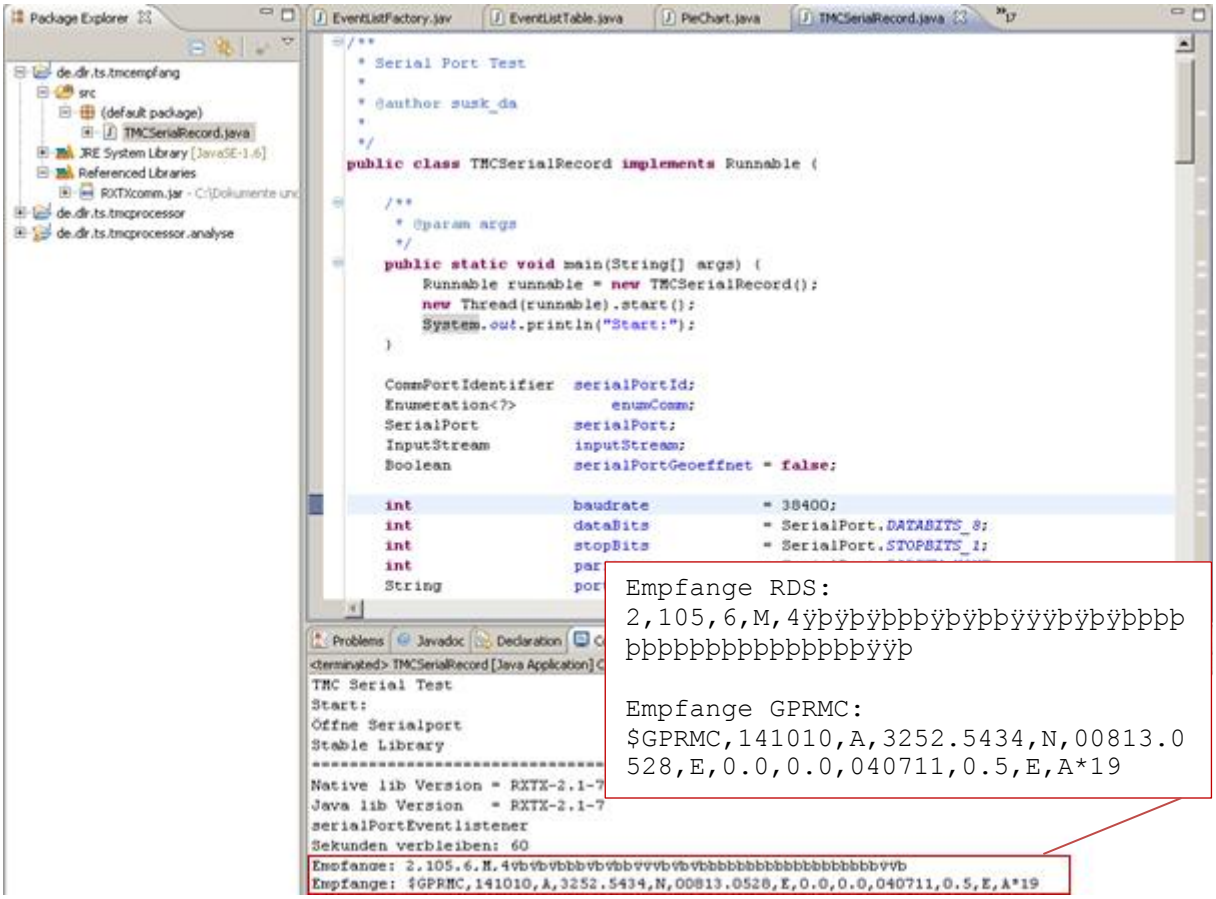


Abbildung 33: Serial Port RxTx Beispiel

Das in der Abbildung 33 geöffnete Konsolenfenster zeigt in den letzten zwei Zeilen den Inhalt der Protokolle (NMEA-0183 und GNS 3.0), die an das Programm übergeben werden. Die National Marine Electronics Association (NMEA) ist eine nicht profitorientierte Vereinigung, die sich für die Ausbildung und den Fortschritt der Elektronikindustrie in der Marine einsetzt. [2] In diesem Rahmen wurde ein einheitlicher Standard geschaffen, der die Abbildung von

[32] Rx = Reciever = Empfänger, Tx = Transmitter = Sender
[33] Application programming interface (API) ist eine Programmierschnittstelle

verschiedenen GPS-Daten auf unterschiedlichsten Geräten sicherstellen soll. [2] Das Resultat ist das NMEA-0183 Protokoll, das auch bei dem Empfänger von Navilock verwendet wird.

Um den Rahmen dieser Arbeit nicht zu überschreiten, soll an dieser Stelle der in der Abbildung 33 „Serial Port RxTx Beispiel“ empfangende Mindestdatensatz des NMEA-Protokolls aber keine Erweiterung beschrieben werden.

Der kleinste zu erwartende Datensatz ist nach diesem Protokoll der GPRMC-Datensatz, der „Global Position Recommended Minimum Sentence C“. [44] Der in Abbildung 34 gezeigte bzw. empfangene String würde nach dem Protokoll, wie folgt zu interpretieren sein.

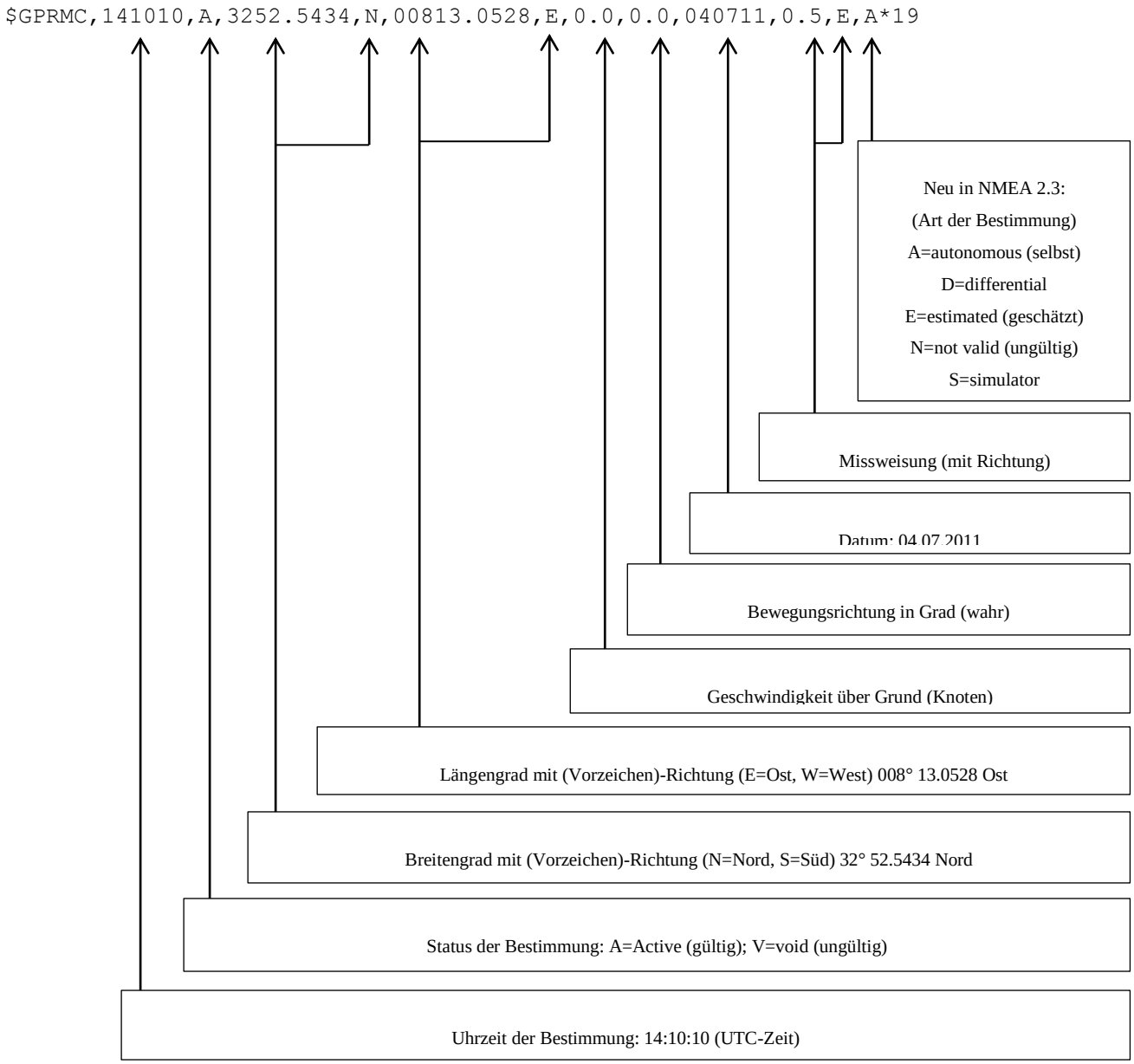


Abbildung 34: NMEA 0183 Protokoll

Wie an dem empfangenen Datensatz zu sehen ist, beginnen die Meldungen mit einem „$“ Zeichen, anschließend folgt durch Kommata getrennt der eigentliche Inhalt, der aus der

Abbildung 34 abzulesen ist. Schwierigkeiten könnten sich beim Verständnis des vorletzten Informationsgehaltes ergeben, der Missweisung. Diese beschreibt den Unterschied zwischen magnetischer und geografischer Position in Metern in diesem Format und spielte früher bei der Kompassnavigation eine Rolle. Ferner schreibt das Protokoll vor, dass die Daten im ASCII-Format[34] gesendet werden, was die Lesbarkeit im Konsolenfenster erklärt.

Der empfangene RDS-Datensatz „2,105,6,M,4ÿþÿþÿþþþÿþÿþþÿÿÿþÿþÿþþþþþþþ-þþþþþþþþþþþÿÿþ“ ist hingegen nicht lesbar. Dies ist der Tatsache geschuldet, dass das GNS 3.0 Protokoll Daten verschlüsselt sendet. Die Firma „Global Navigation System“ hält die Entschlüsselung ihres Protokolls geheim. Der Empfang von TMC mittels GNS-Protokoll ist so weit verbreitet, dass man von einem de facto Standard spricht. Um die ankommenden Daten übersetzen zu können, muss man sich mit der Firma in Verbindung setzen. Nach der Unterzeichnung einer Verschwiegenheitserklärung, wird die aktuelle „GPSTMC-API“, hier in der Version 4.3, zur Verfügung gestellt. Diese in C++ geschriebene API ermöglicht es, das GNS 3.0 Protokoll im „BT-465 UTE – Bluetooth/USB GPS/TMC“ Empfänger anzusprechen. Ohne die zuvor erwähnte Erklärung verletzen zu müssen, kann zur API folgendes hinzugefügt werden.

Die Funktionalität der GPSTMC-API wird in einer dynamischen Verbindungsbibliothek eingekapselt. Somit werden wichtige Funktionen exportiert, in dem Meldungen an ein zuvor definiertes Fenster übergeben werden. Solche Meldungen können eingegangene TMC-Meldungen oder geänderte Frequenzen des Tuners sein. Die gelieferte Release Struktur enthält dabei die Dokumentation der API „GpsTmcApi.pdf“ mit seinen Funktionen, eine „gpstmc.h“ Datei der Header, welcher eingebunden werden muss, um die Funktionen nutzen zu können und die dazugehörige „gpstmx.dll“ Datei. Die „dynamic link library“ fungiert innerhalb des Programms als Bibliothek. Mittels dieser drei Dateien ist es möglich, ein eigenes C++-Programm zu schreiben, welches den TMC-Empfänger steuert. Die Abbildung 35 „GpsTmcReceiver“ zeigt das aus Abschnitt 3.2.2. „Datenquellen“ bekannte Userinterface, mit dem Radiosender gefunden wurden, die TMC ausstrahlen. Die Bedienung wurde unter dem erwähnten Abschnitt bereits hinreichend erklärt und soll jetzt mit Auszügen aus dem zugehörigen Quellcode komplettiert werden.

[34] American Standard Code for Information Interchange (ASCII) ist der Amerikanische Standard Code für Informationsaustausch [2]

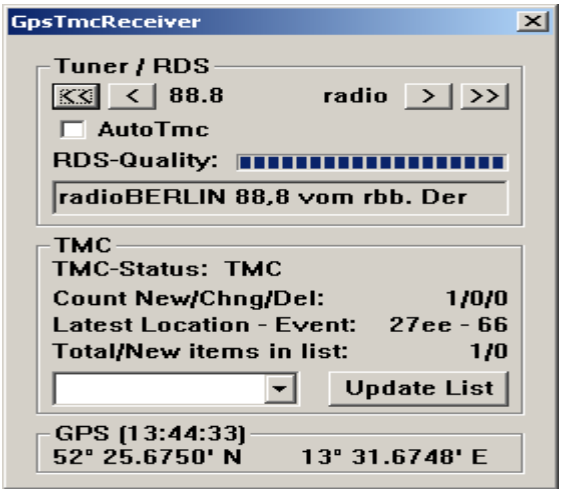


Abbildung 35: GpsTmcReceiver

Um die abgebildeten Funktionalitäten in den einzelnen Bereichen Tuner/RDS, TMC und GPS sicherzustellen, müssen folgende Funktionen der API genutzt werden. [45]

Tabelle 7: Wichtige Meldungen und Funktionen der GNS-API [45]

Tuner/RDS	
Methode	**Beschreibung**
GPSTMC_API BOOL GpsTmc_ControlTuner(DWORD *direction*, int *frequency*);	Diese Funktion ermöglicht die Einstellung der Frequenzen des Tuners. (Verbunden mit Pfeilknöpfen)
GPSTMC_API BOOL GpsTmc_GetTunerFrequency(int *piFrequency);	Setzen der Tuner Frequenzen.
WM_GPSTMC_RDS_PS_CHANGED *wPiCode* = (WORD)*wParam;*	Nachricht wird gesendet, wenn der Name des Radiosenders sich ändert.
WM_GPSTMC_RDS_AF_CHANGED	Wenn alternative Frequenzen sich ändern. (siehe Dienste Abschnitt 3.1.4)
GPSTMC_API BOOL GpsTmc_GetRdsStation(DWORD **pdwPiCode*, WCHAR *awcPs*[9]);	Gibt den RDS-Stations Namen zurück und dessen ID. (PI-Code)
GPSTMC_API int GpsTmc_GetRdsHealth();	Gibt die RDS-Qualität von 0 bis 10 als int. Zurück.

GPSTMC_API BOOL GpsTmc_GetRadiotext(WCHAR *awcRT*[65]);	Gibt den Radiotext zurück. (siehe Textfeld)

TMC

Methode	Beschreibung
WM_GPSTMC_TMC_NEW_DATA *wLocation* = (WORD)*wParam; hTmc* = (HTMC)*lParam;*	Wird gesendet, wenn neue TMC-Daten empfangen wurden. (Anzeigen in der UI von Location-Code und Event und Erhöhung des Zählers)
WM_GPSTMC_TMC_CHANGED *wLocation* = (WORD)*wParam; hTmc* = (HTMC)*lParam;*	Wird gesendet, wenn neue TMC-Daten empfangen wurden, die sich auf eine ältere beziehen, diese verändern. (Anzeigen in der UI von Location-Code und Event und Zähler erhöhen)
WM_GPSTMC_TMC_DELETE *wLocation* = (WORD)*wParam; hTmc* = (HTMC)*lParam;*	Wird gesendet, wenn neue TMC-Daten empfangen wurden, die sich auf eine ältere beziehen, diese löschen. (Löschen aus der Liste in der UI von Location-Code und Event und Erhöhung des Zählers)
GPSTMC_API BOOL GpsTmc_ReadTmcData(HTMC hTmc, PTMC_USERDATA pTmcData);	Die Funktion gibt „wahr“ zurück, wenn der als zweites zu übergebene Pointer auf die neuen TMC-Daten zeigt. Der Datentyp ist dabei vom User selbst anzulegen.

GPS

Methode	Beschreibung
GPSTMC_API DWORD GpsTmc_ReadRawNMEA(char **buf*, int *len*);	Liest die GPS-Daten nach NMEA aus

Tabelle 7 zeigt die wichtigsten Meldungen der bereitgestellten API, die genutzt wurden, um das Userinterface zu programmieren. Die Tabelle ist in die gleichen Abschnitte gegliedert wie die Oberfläche. Die linke Spalte zeigt den Namen der Funktion mit seinen Übergabeparametern, wie sie auch in der Dokumentation der API zu finden sind. Die rechte

Spalte erklärt dessen Funktionalität und bringt sie teilweise in Zusammenhang mit der Benutzeroberfläche.

Die Nutzung der API ist sehr komfortabel und lässt das weitere Bestreben unnötig erscheinen, den TMC-Empfänger selbst über RxTx auszulesen und dessen RDS-Gruppen nach den in Kapitel 3. „Grundlagen“ definierten Kriterien zu interpretieren.

Um ein besseres Verständnis zu schaffen, soll am Beispiel des TMC-Empfangs die Arbeitsweise der API näher erklärt werden. Dafür werden Kenntnisse in C++ Programmierung vorausgesetzt.

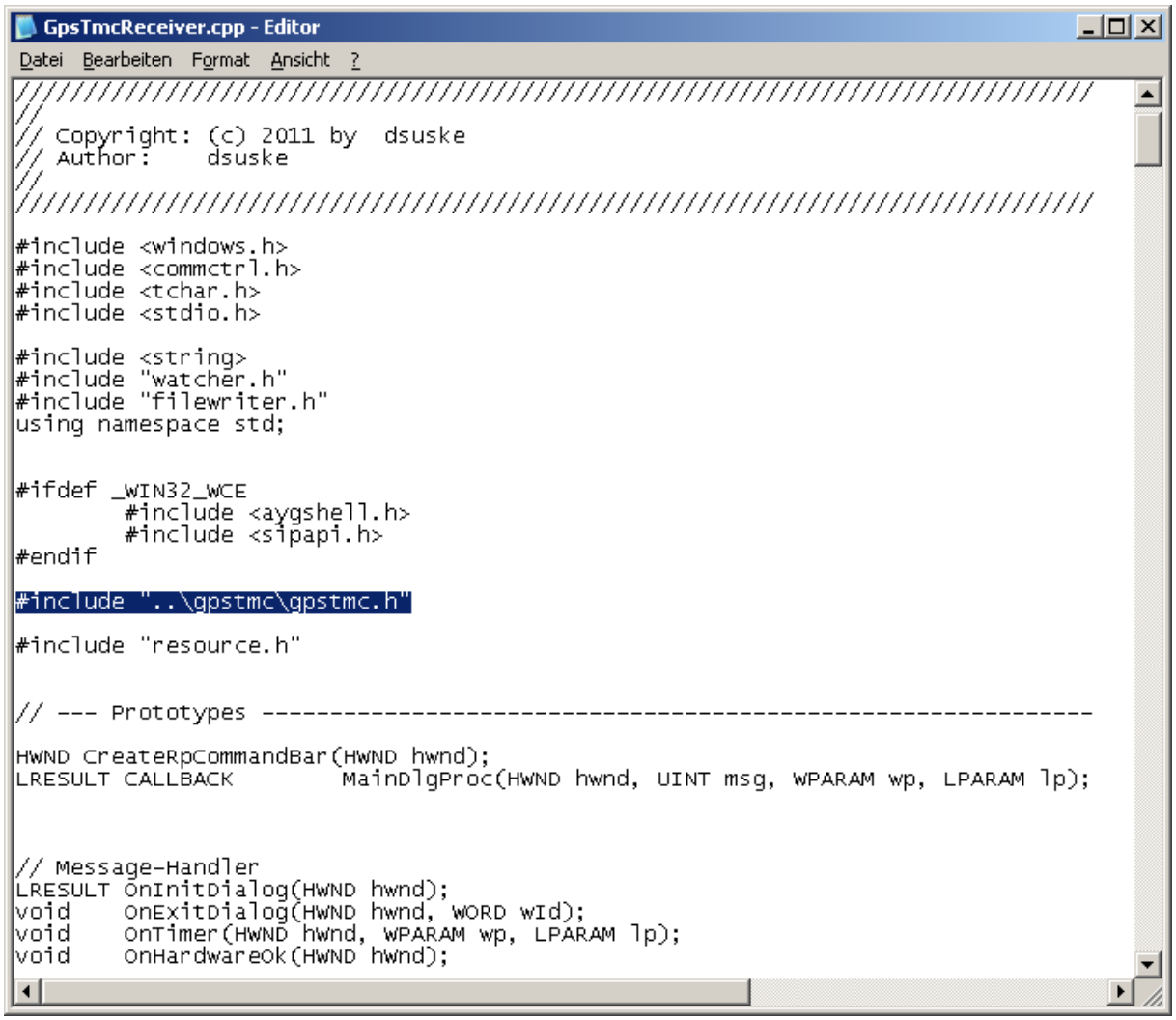


Abbildung 36: GPSTMCReceiver.cpp Header

Die Abbildung 36 oben zeigt den Beginn des Quellcodes, des Programms unter Abbildung 35 „GpsTmcReceiver“. Der hier blaumarkierte Bereich zeigt das Einbinden der mitgelieferten Header-Datei „gpstmc.h“. Neben dem Standard Headers, wie z.B. „stdio.h“o der „windows.h“, werden auch die selbstgeschrieben Header „watcher.h“ und „filewriter.h“ in das

Programm eingebunden. Der „watcher.h“ enthält einen File-Listener, der reagiert, wenn die Datei „config.txt“ verändert wird. Die Konfigurationsdatei bietet dabei dieselbe Funktionalität wie das Userinterface. Da, wie aus Kapitel 4 „Anforderungsanalyse“ bekannt ist, die Module als Services auf dem Server laufen, muss es auch hier eine Einstellmöglichkeit geben, die vorhandenen Funktionen des erstellten Programmes zu bedienen.

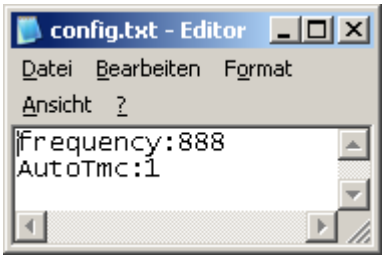


Abbildung 37: GpsTmcReceiver Config.txt

Die in Abbildung 37 „GpsTmcReceiver Config.txt“ zu sehenden Einstellungsmöglichkeiten werden bei jedem Start des Programmes initialisiert und bei einer Veränderung der Daten neu gelesen. In diesem Fall wird die Frequenz des Tuners auf 88.8MHz eingestellt und AutoTmc aktiviert. Das hat zur Folge, dass der Empfänger automatisch eine neue Frequenz suchen würde, wenn der Empfang bei 88.8MHz sich verschlechtert. Die bereits zu Abbildung 10 auf Seite -20- beschriebene Empfangsskala von 0 bis 10 müsste dabei unter 5 fallen, um diesen Prozess auszulösen. Der „filewriter.h“ stellt die Funktionalität bereit, um die neue TMC-Meldung in der Datei „TMCUserData.txt“ abspeichern zu können.

Nach der Initialisierung und der Erstellung eines neuen EventHandlers „HTMC“, der von der Klasse Handle erbt, kann dieser dem Fenster und den Funktionen der API übergeben werden und somit auf die Ereignisse reagieren. Wie diese prozeduralen Abläufe in ihrer Kette agieren können, soll im Folgenden besser verdeutlicht werden.

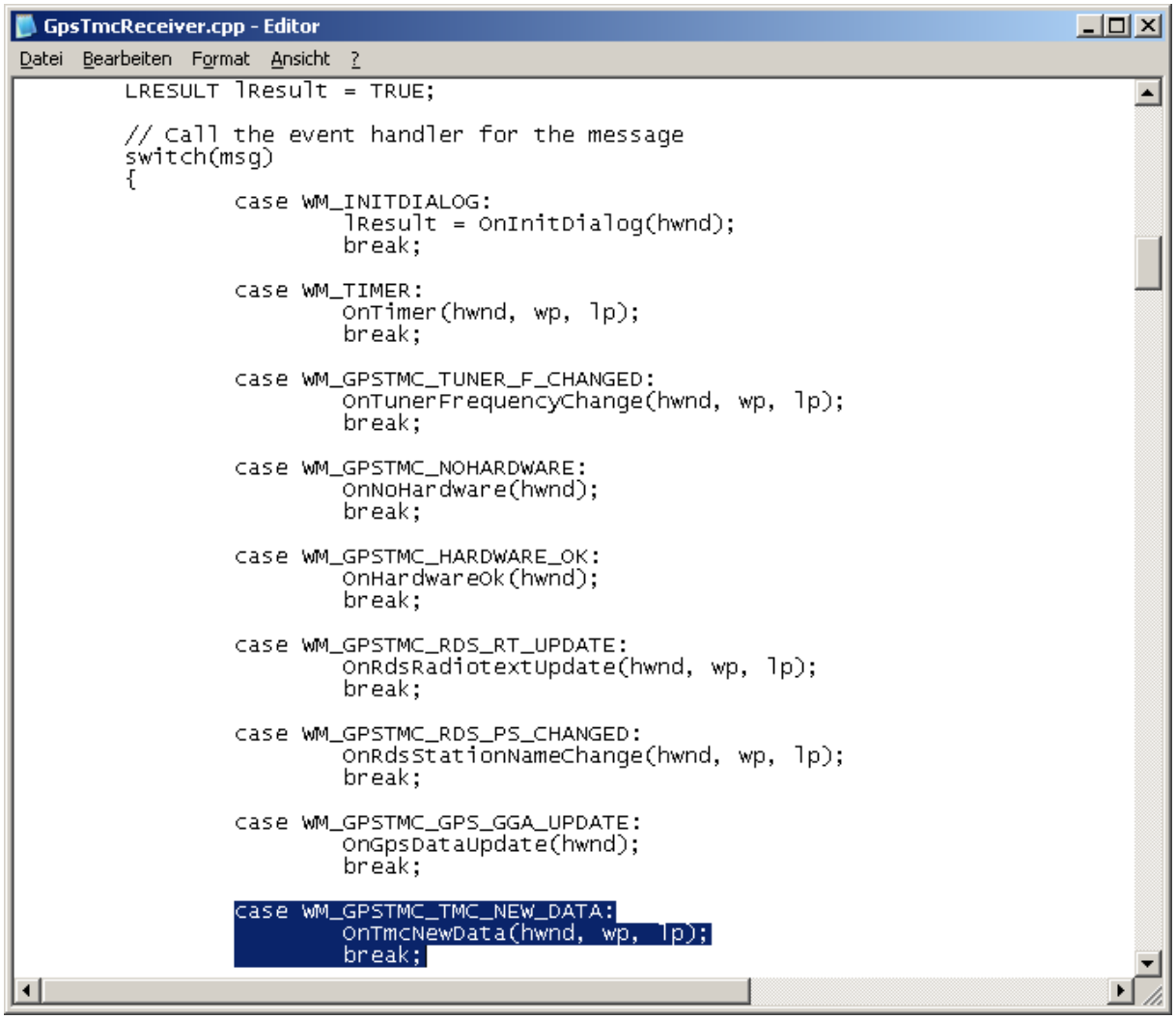


Abbildung 38: GpsTmcReceiver.cpp Event-Handler

Das aus der Tabelle 7 „Wichtige Meldungen und Funktionen der GNS-API“ bekannte Ereignis „WM_GPSTMC_TMC_NEW_DATA“ wird vom registrierten Event-Handler in einer Switch-Case-Anweisung ausgewertet und ruft infolgedessen die Methode „OnTmcNewData(hwnd, wp, lp)“ auf, wie die Abbildung 38 zeigt. Um das Ankommen des Events sicherzustellen, muss der Event-Handler bei der Initialisierung des Tuners mit der Methode „GpsTmc_Init()“ übergeben werden. So ist bekannt, wer die von der API generierten Meldungen erhalten soll.

Des Weiteren werden der Port, Baudrate[35] und der Modus an die Initialisierungsmethode übergeben. Die ebenfalls in der „gpstmc.h“ Datei enthaltenden Modi zeigt die Abbildung 39.

[35] Baudrate ist ein Maß für die Schrittgeschwindigkeit bei der Übertragung in der Nachrichtentechnik. Dabei entspricht 1 Baud der Geschwindigkeit, wenn 1 Zeichen pro Sekunde übertragen wird.

gpstmc.h - Editor

Datei Bearbeiten Format Ansicht ?

```
//
#define EVAL_RDSTIME     0x000100 // evaluates the RDS date&time
#define EVAL_RDSTEXT     0x000200 // evaluates RDS text
#define EVAL_AF          0x000400 // evaluates Alternative frequencies
#define EVAL_PS          0x000800 // evaluates station name
#define EVAL_TMC         0x001000 // evaluates TMC
#define EVAL_TMCRAW      0x010000 // evaluates raw TMC
#define EVAL_GGA         0x002000 // evaluates health GPS & time & position
#define EVAL_RMC         0x004000 // evaluates RMC
#define EVAL_VTG         0x008000 // evaluates VTG
#define EVAL_GSV         0x040000 // evaluates GSV
#define EVAL_NMEARAW     0x100000 // evaluates raw NMEA, disables EVAL_GGA
                                  // EVAL_RMC, EVAL_VTG, EVAL_GSV

//
#define EVAL_SEARCHFRQ   0x020000 // Convey all frequency-steps during a search
//
#define EVAL_GPSALL      0x04E000 // eval all of GPS !
#define EVAL_RDSALL      0x001F00 // eval all of RDS - TMC!
#define EVAL_ALL         0x04FF00 // eval all RDS and GPS !
```

Abbildung 39: Modi der GPSTMCAPI

Der im Programm übergebene Wert EVAL_ALL ermöglicht die Erreichbarkeit aller Komponenten der API wie GPS, RDS und TMC und deren komfortables Auslesen.

Zurück zu der zuvor betrachteten Verarbeitungskette. In der Switch-Case-Anweisung wurde das eingehende Ereignis interpretiert und an die Methode „OnTmcNewData()“ weiter geleitet.

GpsTmcReceiver.cpp - Editor

Datei Bearbeiten Format Ansicht ?

```
void OnTmcNewData(HWND hwnd, WPARAM wp, LPARAM lp)
{
        TMC_USERDATA TmcData;

        BOOL bOk = GpsTmc_ReadTmcData((HTMC)lp, &TmcData);
        if (bOk)
        {
                DlgPrintf(hwnd, IDS_TMC_LATEST, _T("%.4hx - %.2hx"),
                        TmcData.wLoc2, TmcData.wEvents[0]);
                FileWriter::writeToTMCUserData(TmcData);
        }
        else
                SetDlgItemText(hwnd, IDS_TMC_LATEST, _T("? ?  - ?"));


        g_nTmcNew++;
        UpdateTmcCounts(hwnd);
}
```

Abbildung 40: GpsTmcReceiver.cpp OnTmcNewData

Ein Blick in die Methode zeigt die Abbildung 40. Zu Beginn wird ein Objekt des Typs „TMC_USERDATA“ angelegt. (Auf diesen wichtigen Datentyp, der eine TMC-Meldung repräsentiert, wird nach der Beschreibung der Aufruflogik genau eingegangen). Anschließend

wird der Funktion „GpsTmc_ReadTmcData((HTMC)lp, &TmcData)“ die Adresse des zuvor erstellten Objektes und der Event-Handler vom Typ „HTMC“ übergeben. Wenn das Lesen erfolgreich war, also der Boolean „bok“ gleich „true“ ist, wird der neue Location-Code und der Event Code im Userinterface unter Zuhilfenahme der Methode „DlgPrintf()“ dargestellt. Des Weiteren wird die Methode „writeToTMCUserData(TmcData)“ aufgerufen, die die neue Meldung in einer Datei speichert.

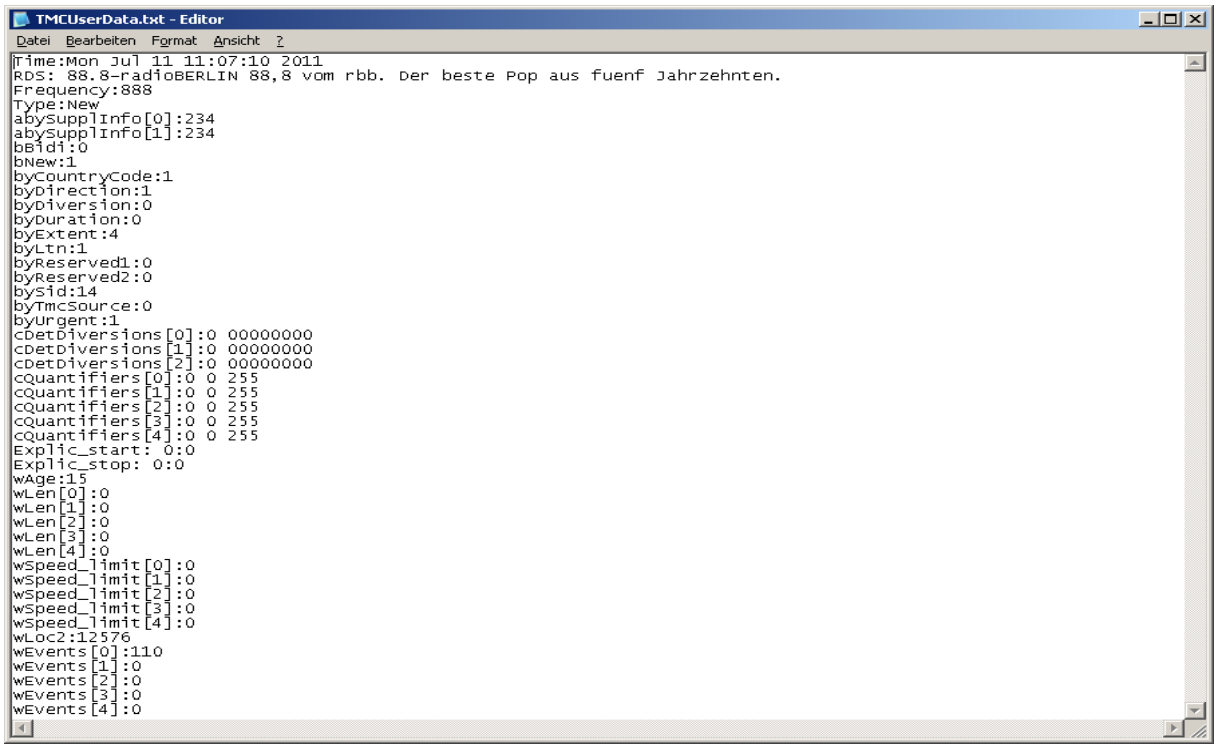


Abbildung 41: TMCUserData

Die Abbildung 41 zeigt, welche Informationen in der Datei „TMCUserData.txt“ gespeichert werden. Die ersten drei Zeilen gehören nicht zum „struct“[36] und spiegeln somit Zusatzinformationen wieder. Hierzu gehören der Zeitstempel, der RDS-String und die Frequenz der TMC-Meldung. Die restlichen Informationen werden aus folgendem „struct“ ausgelesen und in die Datei geschrieben.

[36] Struct ist in C++-Programmierung ein Schlüsselwort, um einen eigenen Datentyp anzulegen.

```
typedef struct tagTMC_USERDATA {
        HTMC hTmc;
        WORD wAge;
        BOOL bNew;
        DWORD dwSort;
        BOOL bBidi;
        QUANTIF cQuantifiers[5];
        DETDIV cDetDiversions[3];
        TIME78 Explic_start;
        TIME78 Explic_stop;
        WORD wLoc2;
        WORD wEvents[5];
        WORD wLen[5];
        WORD wSpeed_limit[5];
        BYTE abySupplInfo[SUPPL_INFO_COUNT];
        BYTE byReserved1;
        BYTE byReserved2;
        BYTE byDirection;
        BYTE byUrgent;
        BYTE byCountryCode;
        BYTE byExtent;
        BYTE byDuration;
        BYTE byDiversion;
        BYTE byLtn;
        BYTE bySid;
        BYTE byTmcSource;
} TMC_USERDATA;
```

Abbildung 42: Struct TMC_USERDATA

Die Abbildung 42 zeigt den neuen Datentyp innerhalb des „Datenimporter“. Ein erstelltes Objekt dieses Datentyps, wird an „GpsTmc_ReadTmcData((HTMC)lp, &TmcData)“ übergeben bzw. dessen Adresse und dann gefüllt. Die einzelnen Typen innerhalb des neuen Datentyps sind dabei aus den entsprechenden ISO Standards ISO 14819-1 bis 14819-3 entnommen. [34] [35] [36] Auf Basis des durch die Grundlagen angeeigneten Wissens sind folgende Informationen von einer TMC-Meldung zu erwarten.

Tabelle 8: TMC_USERDATA

Alter	Die Zeitspanne in Minuten, die diese Aufzeichnung noch gültig ist.
New	Flag, der beschreibt, dass diese Nachricht schon mal gesendet wurde.

Bidi	Flag ist wahr (true) wenn das TMC-Signal in beide Richtung gesendet werden kann.
Quantifiers	Zusatzinformationen zu den Events
DetDiversions	Ein Feld der Größe 3, das mögliche Empfehlungen für alternative Routen enthält.
Explic_start	Ausdrückliche Beginnzeit (UTC) für ein TMC-Ereignis. Wenn keine ausdrückliche Zeit gegeben wird, ist der Wert 0.
Explic_stop	Ausdrückliche EndZeit (UTC) für ein TMC-Ereignis. Wenn keine ausdrückliche Zeit gegeben wird, ist der Wert 0.
LocationCode	Der TMC Location-Code nach (ISO 14819-1), der Ort des Geschehens.
Events	Das Ereignisfeld beinhaltet maximal 5 Events zum zugehörigen Location-Code.
Length	Länge zum zugehörigen Event (z.B. Stau) in km.
Speed_limit	Feld der Größe 5 mit etwaigen Geschwindigkeitsbegrenzungen vor Ort.
SupplInfo	Eine Reihe von Ergänzungsinformationen wie, z.B. zu erwartende Tendenzen über Staulänge.
Direction	Die Richtung als Flag (0 = positiv, 1 = negativ)
Urgent	Dringlichkeit / Wichtigkeit der Meldung
CountryCode	Der Länder-Code wird in RDS-Spezifikation IEC

	62106:2000 für das jeweilige Land definiert.
Extent	Gibt die angrenzenden Segmente als Location-Code und zeigt somit die positive Richtung an.
Duration	Gibt Auskunft über die mögliche Passierdauer.
Diversion	Bezieht sich auf DetDiversion, wenn alternativ Routen vorgeschlagen werden, ist hier die alternative Route hier zu finden.
LTN	Eine vereinbarte Positionstabelle für jeden Service, der Informationen enthält, um den Bereich des Straßensegmentes oder die Punktposition anzuzeigen, in der die Quelle das Probleme definiert hat. Jeder Service hat eine Positions-Tabelle, die durch die Positions-Tabellen-Zahl (LTN) definiert wird.
Sid	Die Service Identifikation wird verwendet, um einen bestimmten TMC-Service von einem Diensterbringer einzigartig zu unterscheiden.
TmcSource	Die Quelle der Mitteilung. Entweder öffentlich (TMC_SRC_PUBLIC) oder bedingter Zugang (TMC_SRC_CA).

Die Tabelle 8 „TMC_USERDATA“ beschreibt den neuen Datentyp, wie er im C++-Programm abgebildet wird. Mit Hilfe dieser Tabelle und der erstellten Datei „TMCuserData.txt“ ist es möglich, die TMC-Meldungen zu interpretieren und zu analysieren. Die Vorgehensweise ist analog zu anderen Methoden, die durch ein Event der API gesteuert werden können, wie beispielsweise „WM_GPSTMC_TMC_CHANGED“. Das Ereignis löst wiederum eine Methode aus, die ein Objekt vom Typ „TMC_USERDATA“ übergeben bekommt und es füllt. Lediglich der Typ der Meldung ist ein anderer, statt „New“ für „Neu“ ist dieser jetzt „Change“ für „Ändern“ bzw. „Delete“ für „Löschen“, wenn das entsprechende Ereignis gefeuert wird.

Dieser Abschnitt stellt den Datenimport dar. Die Daten liegen jetzt zur weiteren Verarbeitung im System und können von der Einheit „TMC Process Unit“ weiter genutzt werden. Des

Weiteren suggeriert die unter Abschnitt 4.3 erstellte Betrachtung der Programmiersprachen eine Änderung der Sprache von C++ auf Java. Wie schon in der Problemanalyse formuliert, kann auf Grund einer Zeitersparnis durch eine Bereitstellung einer Programmierschnittstelle, die Sprache innerhalb der TMC-Kette wechseln, nicht aber innerhalb eines Moduls. Dies ist möglich, da die einzelnen Module separat als Service auf dem Server laufen.

Um jedoch trotzdem ein einheitliches Bild innerhalb der TMC-Kette zu schaffen, wurde eine Java Native Übersetzung des C++ „GPSTMCAPI“ versucht. Hierzu kann das Java Native Interface (JNI) genutzt werden, das eine Umsetzung aller Methoden der API benötigt. Das bedeutet, dass nach dem Erstellen der Java Klasse, die alle Methoden und Funktionen der API enthält, diese zunächst kompiliert werden muss. Anschließend soll aus der Java Klasse eine C++-kompatible Header-Datei erstellt werden. Die eigentliche Herausforderung besteht aber in dem Sachverhalt, dass die C++ Methoden in den C++ Klassen ebenfalls zu einer Bibliothek kompiliert werden müssen, wobei die entsprechende Dynamic Link Library (DLL) einen speziellen JNI Header „jni.h“ einbinden muss. Dieser ist in der mitgelieferten DLL nicht enthalten. So wird wiederum die Erstellung einer neuen DLL, die diesen Anforderungen gerecht wird, benötigt. Die Übersetzung aller Methoden und Funktionen sowie das Fehlen des „jni.h“ Headers führte zu der Suche nach einer weiteren Alternative. [46]

Eine zweite und effektivere Variante stellt die Nutzung der Java Native Access (JNA) API dar. Diese Programmierschnittstelle ermöglicht den Zugriff auf DLL-Dateien, die nicht einen entsprechenden Header einbinden bzw. keine weitere Einbindung benötigt. Die Nutzung ist sehr komfortabel. Die API stellt eine Klasse „CLibrary“ zur Verfügung, von der ein erstelltes Objekt mit dem Befehl „Native.loadLibrary(*.dll)“ auf die zu nutzende DLL zeigt. [47] Über das erstellte Objekt sind die Methoden der API erreichbar. Die Problematik ist, dass keine Möglichkeit besteht einen Event-Handler vom Typ Handle über JNA zu registrieren. Bei der API gibt es die Klasse „HWND“, die dies ermöglichen soll. Aber das bei der Initialisierung übergebene Objekt wirft keine Ereignisse. Ferner werden keine Fehler ausgegeben. Auch wenn der TMC-Empfänger auf der übergebenen Frequenz nach der Initialisierung läuft, können neue TMC-Daten nur wahllos ausgelesen werden, indem die entsprechende Methodenübersetzung zufällig ausgeführt und nicht durch ein Event getriggert wird. So ist es möglich, TMC-Daten auszulesen, doch ob eine Meldung vergessen wurde, kann nicht garantiert werden. Dass dies keine Lösung sein kann, ist nachvollziehbar, wenn man bedenkt, dass in einem bestimmten Zeitabschnitt eine unbestimmte Anzahl von TMC-Meldungen zu erwarten sind. Ein konstantes Auslesen pro Sekunde oder noch häufiger könnte sicherstellen, dass jede TMC-Meldung verarbeitet wird. Doch die Performance solch einer Lösung ist

schlecht, vor allem, wenn eine bereitgestellte API eine sehr effiziente Steuerung ermöglicht. Die Registrierungsproblematik des „HWND“ Objektes ist ebenfalls unter dem Bug Fix Report von JNA nachzulesen. [48]

Auf Basis der beschriebenen Probleme mit nativen Übersetzungsbibliotheken wurde davon abgesehen, weitere Bemühungen zu unternehmen, mit einem höheren Zeitaufwand den Datenimporter von TMC in Java zu programmieren. Das mit C++ erstellte Modul funktioniert gut, ist robust und dank der Dokumentation gut zu warten bzw. zu erweitern. Ferner kann der Umfang der gesamten API genutzt werden.

5.2. TMC Process Unit

In diesem Abschnitt wird der weitere Verlauf der Verarbeitungskette vom Datenimporter bis zur persistenten Speicherung gezeigt. Besonders steht die Lesbarkeit der TMC-Daten im Mittelpunkt der Betrachtung. Diese Aufarbeitung der Daten benötigt eine bestimmte Decodierung bzw. Interpretation der Rohdaten und wird in Hinblick auf die Initialisierung- und Bearbeitungsschritte im Folgenden erklärt.

5.2.1. Verbindung zum Datenimporter

Wie aus dem Abschnitt 5.1 bekannt ist, wird jede vom Datenimporter generierte TMC-Meldung in die Datei „TMCUserData.txt“ geschrieben, die automatisch im Ausführungspfad erstellt wird. Das Modul „TMCProcessor“ reagiert über einen Listener auf die Veränderung der Datei und liest die neue Meldung aus. Die Klasse „TMCFileListener“ wird als Thread[37] gestartet, um diese Interaktion für jede Nachricht sicherzustellen und die Arbeit der Hauptroutine nicht zu unterbrechen.

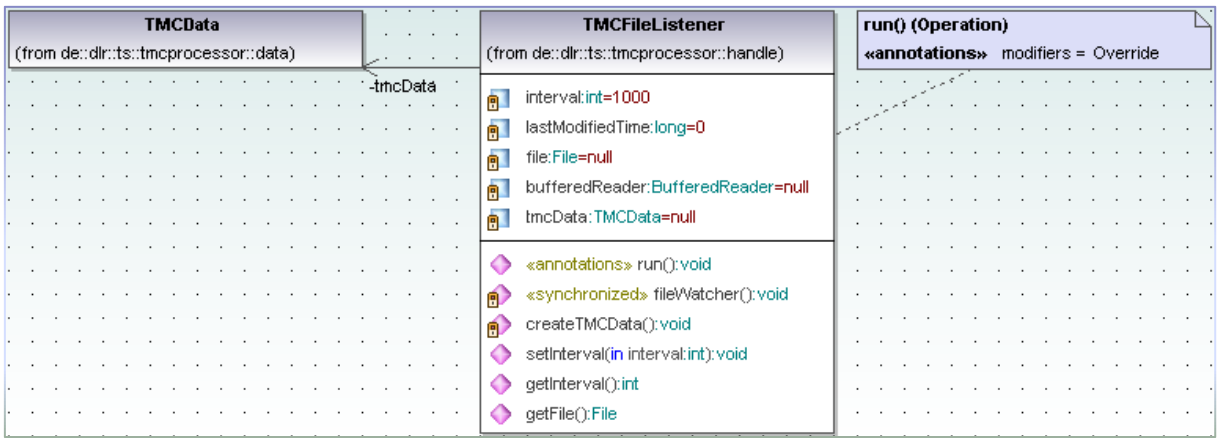


Abbildung 43: UML TMCFileListener [49]

Die Abbildung 43 „UML TMCFileListener“ zeigt den Aufbau der Klasse in einem UML[38]-Diagramm. Die Methode „fileWatcher()“ wird in diesem Thread aufgerufen und überprüft, ob sich die Datei verändert hat. Wenn dem so ist, werden mit der Methode „createTMCData()“ die neuen TMC-Daten in einem Objekt der Klasse „TMCData“ gespeichert.

Das Objekt der Klasse „TMCData“ ist dabei eine Java Übersetzung des unter C++ beschriebenen Datentyps „TMC_USERDATA“. Das erstellte Objekt soll nach seiner

[37] Thread ist eine sequentielle Abfolge von Anweisungen. Wichtig ist dabei, dass mehrere Thread‘s parallel laufen können.
[38] Unified Modeling Language (UML) ist eine grafische Modellierungssprache, die genutzt wird, um komplexe System zu beschreiben. [11]

Erstellung decodiert in die Datenbank geschrieben werden, um die Lesbarkeit zu erhöhen und eine Weiterverarbeitung zu vereinfachen. Das ist wie folgt zu verstehen.

5.2.2. Decodierungsinitialisierung

Das Objekt „tmcData“ repräsentiert eine TMC-Meldung und stellt alle Informationen über diese bereit. Die Ortsinformation beispielsweise, könnte mit dem Methodenaufruf „tmcData.getwLoc2()“ abgerufen werden. Zur Interpretation der Zahl benötigt man die Location-Code-Liste innerhalb des Programms. Um dies zu bewerkstelligen wird zum Beginn des Programmes die CSV-Datei „LCL10.1.D-110221.csv“ ausgelesen. Diese enthält die aktuelle Location-Code-List, wie sie ausschnittsweise in Abbildung 13 „Location-Code-List“ gezeigt wird. Dabei werden die Spalten in einer Klasse „LocationCodeData“ abgebildet, für jede Zeile aus der Datei ein Objekt vom Typ „LocationCodeData“ erstellt und in der HashMap „hashMapLocationCode“ gespeichert.

Eine HashMap ist ein Datentyp, der einen Schlüssel auf einen Wert abbildet, also miteinander verknüpft. Zum Aufbau dieser Datenstruktur innerhalb der HashMap wird eine Hashfunktion[39] genutzt, die immer konstante Zugriffszeiten garantiert. Aus diesem Grunde bietet sich der Datentyp HashMap für diesen Gebrauch besonders an. Der Schlüssel der HashMap ist der Location-Code selbst, da dieser eineindeutig ist. Der zu speichernde Wert ist das Objekt, das wiederum jede Spalte in dieser Zeile referenziert. [50]

[39] Die Steuerfunktion (Hashfunktion) bildet eine deterministisch große Datenmenge auf eine kleine ab. [74]

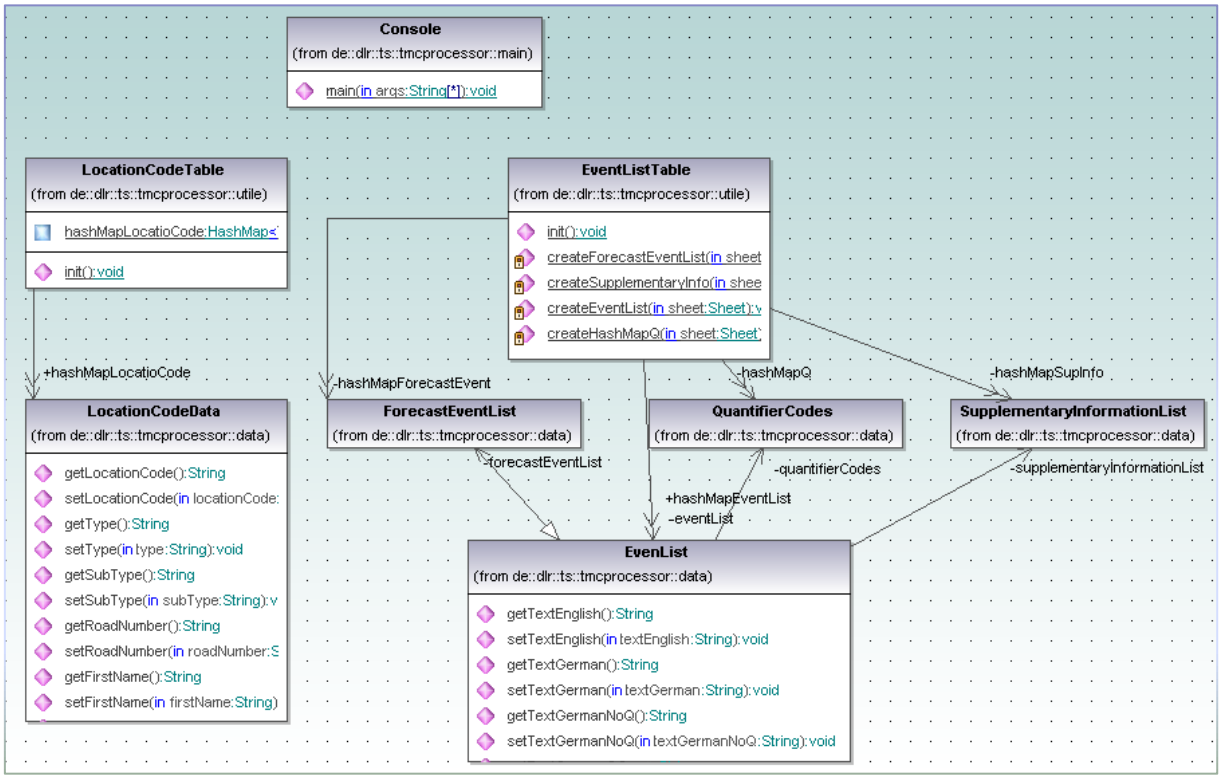


Abbildung 44: UML Location-Code & Event-List [49]

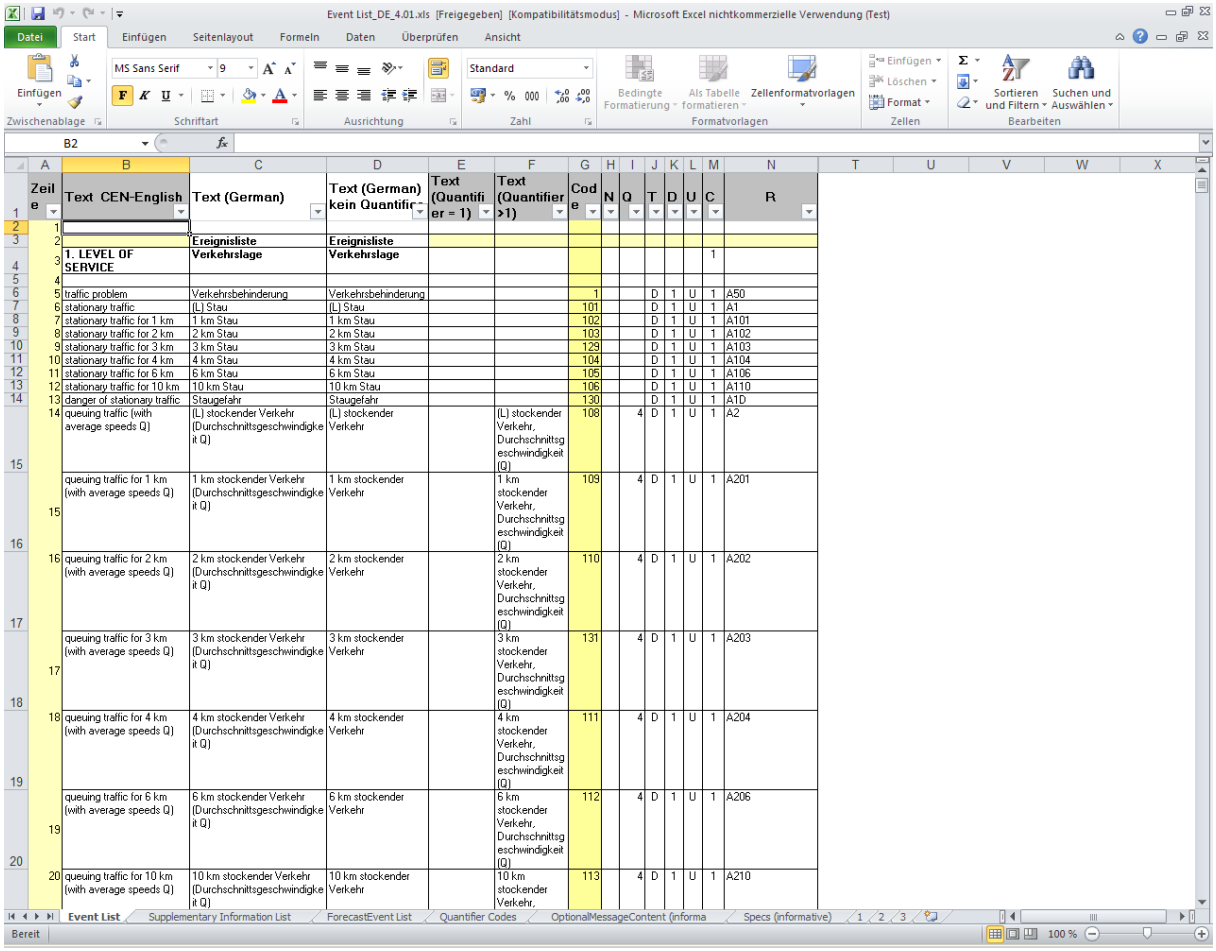
Die Abbildung 44 „UML Location-Code & Event-List“ zeigt links den zuvor beschrieben hierarchischen Aufbau der Klassen „LocationCodeTable“ und „LocationCodeData“. Wie in der Abbildung zu sehen, können die Variablen durch die entsprechenden Methoden gesetzt oder ausgelesen werden. Ferner werden nicht alle Methoden der jeweiligen Klassen gezeigt, da vor allem die Abhängigkeiten zueinander hier im Vordergrund der Betrachtungen stehen sollen. Die Klasse „EventList“ (rechts unten in der Abbildung) wird im gleichen Maße über die Init-Methode der Klasse „EventListTable“ erstellt. Das bedeutet, dass die Klasse „EventList“ die Spalten der Excel-Datei abbildet und die Klasse „EventListTable“, die Objekte zu jeder Zeile. Somit ist über „hashMapEventList“ ein Objekt von „EventList“ über den Event-Code referenzierbar.

Das hier vorgestellte Konzept liegt folgender übergeordneter Struktur zugrunde. Die „Table-Klassen“ verwalten die HashMaps, die wiederum Objektinterpretationen der jeweiligen Dateien enthalten. Dieser einfache Aufbau sichert die Verwaltung der internen Datenstrukturen an einem zentralen Punkt und ermöglicht einen intuitiven Zugriff auf die Informationen in den Dateien. [51][S127]

Bei der Betrachtung des UML-Diagramms in Abbildung 44 fällt auf, dass die Klassen „ForecastEventList“, „QuantifierCodes“ und „SupplementaryInformationList“ von der Klasse

„EventList“ erben, um jeweils auch eine Eventliste zu repräsentieren. Die Konsequenz ist, dass diese Objekte auch in der HashMap „hashMapEventList“ gespeichert werden können.

Die Schaffung dieser hierarchischen Struktur ist effizient und sichert einen einfachen Umgang mit den Daten. Dies wird deutlich, wenn die Excel-Datei „Event List_DE_4.01.xls“ genauer betrachtet wird.



Zeile	Text CEN-English	Text (German)	Text (German) kein Quantifier	Text (Quantifier = 1)	Text (Quantifier >1)	Code	N	Q	T	D	U	C	R
1													
2		Ereignisliste	Ereignisliste										
3	1. LEVEL OF SERVICE	Verkehrslage	Verkehrslage									1	
4													
5	traffic problem	Verkehrsbehinderung	Verkehrsbehinderung			1			D	1	U	1	A50
6	stationary traffic	(L) Stau	(L) Stau			101			D	1	U	1	A1
7	stationary traffic for 1 km	1 km Stau	1 km Stau			102			D	1	U	1	A101
8	stationary traffic for 2 km	2 km Stau	2 km Stau			103			D	1	U	1	A102
9	stationary traffic for 3 km	3 km Stau	3 km Stau			129			D	1	U	1	A103
10	stationary traffic for 4 km	4 km Stau	4 km Stau			104			D	1	U	1	A104
11	stationary traffic for 6 km	6 km Stau	6 km Stau			105			D	1	U	1	A106
12	stationary traffic for 10 km	10 km Stau	10 km Stau			106			D	1	U	1	A110
13	danger of stationary traffic	Staugefahr	Staugefahr			130			D	1	U	1	A1D
14	queuing traffic (with average speeds Q)	(L) stockender Verkehr (Durchschnittsgeschwindigkeit Q)	(L) stockender Verkehr		(L) stockender Verkehr, Durchschnittsgeschwindigkeit (Q)	108		4	D	1	U	1	A2
15	queuing traffic for 1 km (with average speeds Q)	1 km stockender Verkehr (Durchschnittsgeschwindigkeit Q)	1 km stockender Verkehr		1 km stockender Verkehr, Durchschnittsgeschwindigkeit (Q)	109		4	D	1	U	1	A201
16	queuing traffic for 2 km (with average speeds Q)	2 km stockender Verkehr (Durchschnittsgeschwindigkeit Q)	2 km stockender Verkehr		2 km stockender Verkehr, Durchschnittsgeschwindigkeit (Q)	110		4	D	1	U	1	A202
17	queuing traffic for 3 km (with average speeds Q)	3 km stockender Verkehr (Durchschnittsgeschwindigkeit Q)	3 km stockender Verkehr		3 km stockender Verkehr, Durchschnittsgeschwindigkeit (Q)	131		4	D	1	U	1	A203
18	queuing traffic for 4 km (with average speeds Q)	4 km stockender Verkehr (Durchschnittsgeschwindigkeit Q)	4 km stockender Verkehr		4 km stockender Verkehr, Durchschnittsgeschwindigkeit (Q)	111		4	D	1	U	1	A204
19	queuing traffic for 6 km (with average speeds Q)	6 km stockender Verkehr (Durchschnittsgeschwindigkeit Q)	6 km stockender Verkehr		6 km stockender Verkehr, Durchschnittsgeschwindigkeit (Q)	112		4	D	1	U	1	A206
20	queuing traffic for 10 km (with average speeds Q)	10 km stockender Verkehr (Durchschnittsgeschwindigkeit Q)	10 km stockender Verkehr		10 km stockender Verkehr,	113		4	D	1	U	1	A210

Abbildung 45: Aufbau Event-List

Die Abbildung 45 „Aufbau Event-List“ soll lediglich auf die Taps der Excel-Datei hinweisen. Die ersten vier stellen Informationen für ein Ereignis bereit und müssen sinnvoll in den „TMCProcessor“ übertragen werden. Die beiden anderen Taps „OptionalMessageContent (informative)“ und „Specs (informative)“ erklären Aufbau, Inhalt und Abhängigkeiten der relevanten vier Taps.

Um einen einheitlichen Zugriff über eine HashMap auf alle Informationen der Excel-Datei sicherzustellen, werden über die bereits erwähnte Vererbungshierarchie, Objekte der

einzelnen Taps in der HashMap „hashMapEventList“ gespeichert. Zum Auslesen der Daten bzw. zum Erstellen der einzelnen Objekte wurde die JXL-API[40] genutzt.

```
File inputWorkbook = new File(ConfigHandler.getPathToEventList());
Workbook w;
try {
    w = Workbook.getWorkbook(inputWorkbook);

    // Loop over column and lines for Quantifier Codes
    Sheet sheet = w.getSheet(3);
    createHashMapQ(sheet);
```

Abbildung 46: JXL Beispiel

Die Abbildung 46 „JXL Beispiel“ zeigt den relevanten Funktionsumfang der API bei der Nutzung des „TMCProcessors“. Nach der Erstellung einer Datei (File) kann ein Workbook erstellt werden. Dieses Arbeitsbuch entspricht der Excel-Datei und kann nun mit dem Sprachumfang der API bearbeitet werden. Die einzelnen Taps beispielsweise, sind über die Methode „getSheet()“ erreichbar. Wie in Abbildung 45 zu sehen, ist der vierte Tap „Quantifier Codes“ und kann infolgedessen mit dem in Abbildung 46 gezeigten Methodenaufruf „w.getSheet(3)“ in dem Objekt „sheet“ gespeichert werden. Die „chreateHashMapQ(sheet)“ verarbeitet diesen Sheet zu Objekten vom Typ „EventList“. Mit den anderen Taps bzw. Sheets wird analog verfahren.

Die Initialisierung der Ausleseprozesse der beiden Dateien wird von der Main-Methode der Klasse „Console“ gestartet. Nach diesem Prozess liegen die von der BASt herausgegebenen Dokumente bzw. dessen Inhalte als Objekte in den jeweiligen HashMaps vor. Nach dem Aufbau der internen Datenstrukturen können diese genutzt werden, um die vom Datenimporter bereitgestellte TMC-Meldung wie folgt zu interpretieren.

5.2.3. Decodierung

```
String lat;
String lon;

if (LocationCodeTable.hashMapLocatioCode.containsKey(tmcData.getwLoc2())) {

    LocationCodeData locationCodeData = LocationCodeTable.hashMapLocatioCode.get(tmcData.getwLoc2());
    lat = locationCodeData.getLat();
    lon = locationCodeData.getLon();
} else {
    lat = "Not In LCList";
    lon = "Not In LCList";
}
```

Abbildung 47: Location-Code Decode

[40] Java Excel API (JXL-API)

Die Abbildung 47 „Location-Code Decode“ zeigt, wie mit den zuvor erstellten HashMaps gearbeitet werden kann. Die Klasse „LocationCodeTable“ verwaltet die bereits bekannte statische HashMap „hashMapLocationCode“. Der Aufruf der HashMap ist somit auch ohne ein Objekt der Klasse möglich.

Zu Beginn wird mit der Methode „containsKey()“ kontrolliert, ob der Location-Code in der Map enthalten ist bzw. ob zu diesem Schlüssel ein Objekt vorhanden ist. Wenn dem so ist, wird das Objekt vom Typ „LocationCodeData“ aus der Datenstruktur geholt. Dieses enthält nun alle wichtigen Informationen, die auch in der Location-Code-Liste enthalten waren. Der entsprechende Wert, z.B. Längen- und Breitengrad, kann mit den Methoden „getLon()“ und „getLat()“ gelesen werden. Dabei hat jede Variable ihre Getter- und Setter-Methoden. Für die Decodierung der Ereignisse in der Event-Code-List wird analog verfahren: Die entsprechende HashMap nehmen, Objekt geben lassen und über die Methoden die relevanten Werte bestimmen. Dabei wird, neben der geografischen Lage, der vom DLR geforderte Informationsgehalt folgendermaßen aus der TMC-Nachricht entschlüsselt.

```
// Decoded
EventListFactory.decodedDirection(tmcData);
EventListFactory.decodedEvents(tmcData);
EventListFactory.decodedAge(tmcData);
EventListFactory.decodedReplacement(tmcData);
```

Abbildung 48: Event Decode

Der Decodierungsprozess startet bei der Erstellung einer neuen TMC-Meldung im „TMCFileListener“. Wie die Abbildung 48 „Event Decode“ zeigt, wird jede Nachricht mit den entsprechenden Fabrikmethoden[41] decodiert. Die Fabrikmethoden müssen zur Decodierung auf die zuvor beschrieben HashMaps zurückgreifen. Dabei entsprechen die Fabrikmethoden nicht der per Definition erwarteten Verfahrensweise des Entwurfsmusters. Zumal im herkömmlichen Sinne zu erwarten wäre, dass diese Methoden ein Objekt erstellen, welches über entsprechende Methoden die decodierten Informationen bereitstellt.

[41] Fabrikmethode ist ein Entwurfsmuster, das die Erstellung eines Objektes einer Methode zuschreibt und nicht einem Konstruktor.

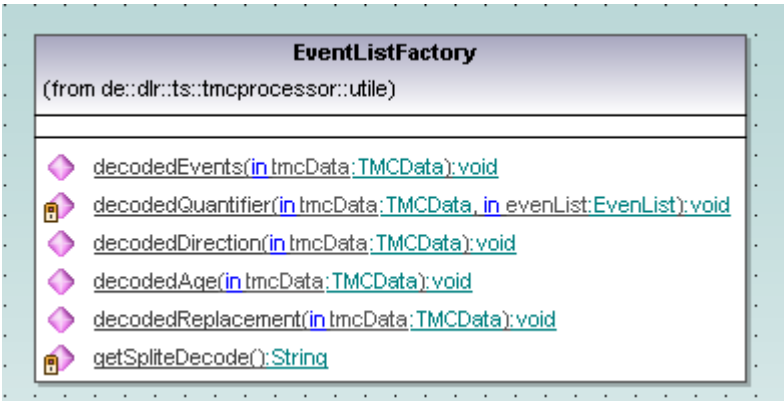


Abbildung 49: EventListFactory [49]

Wie Abbildung 49 „EventListFactory“ aber zeigt, ist der Rückgabetyp „void“ leer. Stattdessen erwarten die Methoden ein Objekt vom Typ „TMCData“. Das Objekt wird nicht neu erstellt. Somit ist „EventListFactory“ als eine Fabrik zu sehen, die das TMC-Objekt um die entsprechenden decodierten Informationen erweitert und es nicht neu generiert. Diese Erweiterungsverfahren werden in dem Buch „Software-Entwurf mit UML 2“ [52]^{S83} beschrieben und werden dem Entwurfsmuster Factory zugeordnet.

Die in Abbildung 48 gezeigte Decodierungsreihenfolge bzw. der Inhalt der Fabrikmethoden soll im Folgendem erklärt werden.

Richtung

Die Richtung des Ereignisses, also welcher Straßenabschnitt betroffen ist, kann aus dem Wert „Direction“ der TMC-Meldung ermittelt werden. Bei einem durch „tmcData.getByDirection()“ geholten Wert 0, ist der Offset positiv. Bei 1 wäre dieser Wert als negativer Offset zu interpretieren.

Ist der Offset positiv, kann nach Tabelle 1 „Location-Code-List Beispiel“ der Wert für den nächsten Location-Code aus der Spalte K geholt werden, wäre er negativ müsste der nächste Location-Code aus Spalte J geholt werden. Aus dem hier abgelesen Location-Code kann mittels der HashMap für den Location-Code wieder das entsprechende „LocationCodeData“ Objekte erzeugt werden, welches wiederum seine eigene geografische Position enthält. Dieses wird solange fortgesetzt, wie der von der TMC-Meldung mitgelieferte Wert „Extent“ es beschreibt. Ist der durch „tmcData.getByExtent()“ bestimmte Wert 2, müsste der, durch den Offset bestimmte Location-Code sich nochmals den nächsten Location-Code geben lassen. Der so bestimmte Abschnitt zeigt die Ausdehnung des jeweiligen Ereignisses und auch dessen Richtung.

Ereignis

Die Decodierung des eigentlichen Ereignisses kann wie bereits beschrieben über die entsprechende HashMap bzw. dessen Objekt vom Typ „EventList“ decodiert werden. Die Bestimmung weiterer Zusatzinformationen ist nach der vorgestellten Datenstruktur denkbar einfach. Das Objekt „eventList“, welches von der HashMap „hashMapEventList“ über den Event-Code zurückgegeben wird, referenziert die Klassen „EventList“, „QuantifierCodes“, „SupplementaryInformationList“ und ForecastEventList“, die aus der Datei „Event List_DE_4.01.xls“ ausgelesen wurden. An einem Beispiel zeigt sich die Einfachheit des Aufrufs.

```
private static void decodedQuantifier(TMCData tmcData, EvenList evenList) {
    if (evenList.getQuantifierCodes() != null) {
        tmcData.setDecodeAppend(getSpliteDecode() + "Speed: " + evenList.getQuantifierCodes().getLabel4Q4Speed()
                + getSpliteDecode() + "Passierdauer: "
                + evenList.getQuantifierCodes().getLabel4Q5Period() + getSpliteDecode() +
                "Länge: " + evenList.getQuantifierCodes().getLabel2AggrievedLayer());
    } else {
        if (tmcData.getByDuration() == null)
            tmcData.setByDuration("0");

        if (!tmcData.getByDuration().equals("0"))
            tmcData.setDecodeAppend(getSpliteDecode() + "Passierdauer: " + tmcData.getByDuration());
    }
}
```

Abbildung 50: QuantifierCodes Beispiel

Die Methode „decodedQunatifier(TMCData tmcData, EventList eventList)“ bekommt das vorgestellte Event-Objekt übergeben, wie in der Abbildung 50 „QuantifierCodes Beispiel“ zu sehen. Wenn nun Zusatzinformationen in dem Tap „QuantifierCodes“ in der besagten XML-Datei vorliegen, wird ein Objekt der Klasse „QuantifierCodes“ erstellt, das über die Methode „getQuantifierCodes()“ referenziert wird. Wenn kein Objekt erstellt wurde, weil in der Liste keine Zusatzinformationen vorlagen, dann ist „null“ als Rückgabewert dieser Methode zu erwarten. Im gleichen Maße werden die Informationen aus den anderen Klassen erstellt. Die Klasse „SupplementaryInformationList“ gibt Zusatzinformationen in Englisch und Deutsch in den Rubriken „Umleitung“, „Fahrzeuge“, „Warnungen“, „Geschwindigkeiten“, „Instruktionen“, „Fahrspurbenutzung“, „Positionen“, „Orte“, „Begründungen“, „Winterliche Verkehrsverhältnisse“, „Anregungen“, „Bezeichner“, „Richtungsangaben“, „Höflichkeitsformeln“ und „Tendenz von Staulängen“. Aussagen über mögliche Verläufe als eine Prognose sind im Objekt der Klasse „ForecastEventList“ gespeichert. Die in den

Rubriken „Erwartete Verkehrslage“, „Wettervorhersage“, „Straßenzustandsvorhersage“, „Umwelt“, „Windvorhersage“, „Temperaturvorhersage“, „Erwartete Zeitverluste und „Erwartete Streichungen“ enthaltenen Informationen werden gegebenenfalls nach gleichem Schema erstellt.

Empfangszeitpunkt & Gültigkeitsdauer

Der Empfangszeitpunkt entspricht dem Zeitstempel der TMC-Meldung und kann über die Methode „tmcData.getTime()“ abgerufen werden. Die Gültigkeitsdauer steht ebenfalls in der Meldung und wird mit „tmcData.getwAge()“ in Minuten bestimmt.

Die Replacement Methode ist im engeren Sinne nicht mehr als Decodierungsmethode zu verstehen. Vielmehr wird die Decodierung auf Lesbarkeit überprüft und gegebenenfalls verbessert.

5.2.4. Speicherung

Die TMC-Meldung und dessen decodierte Informationen werden in die PostgreSQL-Datenbank in die Tabellen „tmc_oneday“ und „tmc_active“ mittels der Klasse „Connector“ gespeichert.

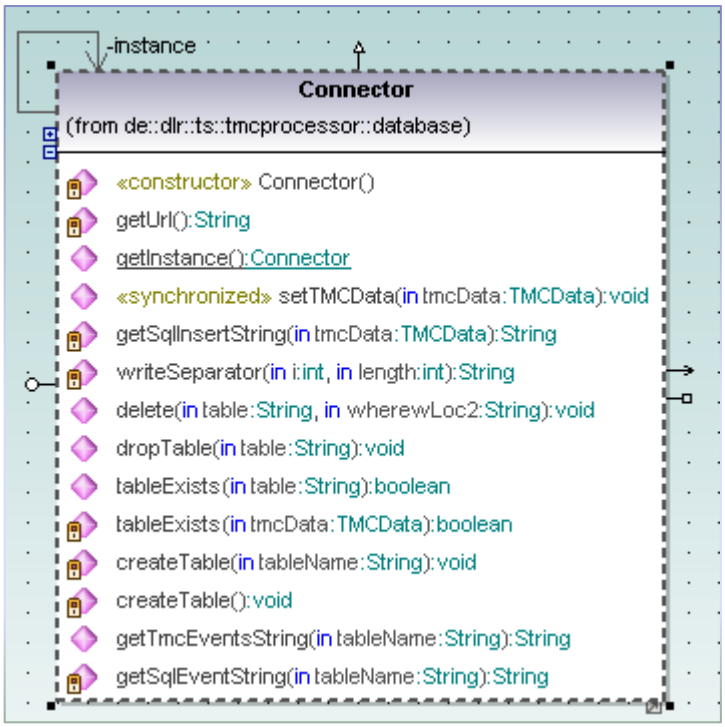


Abbildung 51: JDBC Connector [49]

Die Abbildung 51 „JDBC Connector“ zeigt das UML-Diagramm dieser Klasse. Auffällig ist, dass der Konstruktor der Klasse „private“ und somit nicht für andere Klassen sichtbar ist. Die statische Methode „getInstance()“ dieser Klasse sichert den Zugriff auf das Objekt der Klasse „Connector“. Dieses unter „Singelton“ [51][S101] bekannte Entwurfsmuster garantiert, dass es nur ein Objekt der Klasse geben kann. Diese Maßnahme ist sinnvoll, wenn beabsichtigt wird, dass sich nur ein Objekt einer Klasse, also eine Instance, um den Datenbankzugriff kümmern soll. Dadurch kann verhindert werden, dass innerhalb des Programms mehrere Zugriffe auf die Datenbank von unterschiedlichen Objekten gleichzeitig initialisiert werden. Dies hätte zur Folge, dass unvorhersehbare Schreib- und Lesefehler auftreten können, wenn ein Fehler in der Synchronisation vorliegt. Des Weiteren stellt diese Klasse alle wichtigen Methoden zur Verwaltung der Datenbank mittels „Java Database Connectivity“ bereit. Die JDBC-API bringt dabei den Funktionsumfang mit PostgreSQL-Datenbanken lesen, schreiben, verarbeiten, erstellen und löschen zu können.

Das Schreiben der neuen TMC-Meldung soll an einem Beispiel veranschaulicht werden.

Daten editieren - PostgreSQL 8.4 (129.247.218.180:5432) - tmc - tmc_oneday

Datei Bearbeiten Anzeigen Werkzeuge Hilfe

Keine Begrenzu

	id [PK] character var	tmctime character var	rds character var	wloc2 character var	lat character var	lon character var	wevents character var	wdecode character varying(300)
1	1	Mon Jul 18	Berlin-	10509	52.39710	09.89975	393,0,0,0,0	delimerA2 Hannover-Ost Richtung Hannover-BuchholzdelimerEvent: Gefahr durch defektes F
2	2	Mon Jul 18	Berlin-rad	12543	48.46500	10.11550	1802,0,0,0,	delimerA8 Ulm/Elchingen Richtung LeipheimdelimerEvent: Notrufsäulen ausgefallen delim
3	3	Mon Jul 18	88.8-radio	41170	52.56800	13.31130	406,0,0,0,0	delimerAlll Autobahnzubringer Hamburg Eichborndamm Richtung SeidelstraßedelimerEvent:

Abbildung 52: Tabelle tmc_oneday

Die Abbildung 52 „Tabelle tmc_oneday“ zeigt drei abgelegt Datensätze. Die Tabelle enthält alle Informationen, die aus dem Datenimporter in „TMC_USERDATA“ übergeben wurden (genaue Struktur siehe Tabelle 9), und die bereits mehrfach erwähnte Decodierung. Die Spalte „wloc2“ repräsentiert den LocationCode der TMC-Meldung, z.B. 10509 aus Zeile 1.

Die beiden Tabellen repräsentieren folgendes Datenaufkommen. Die Tabelle „tmc_oneday“ enthält alle TMC-Daten, die durch den Datenimporter geliefert wurden und dessen Decodierung von einem Tag. Die Tabelle „tmc_active“ enthält die aktuellen TMC-Meldungen eines Tages. Hier scheinen sich die Tabellen nicht voneinander zu unterscheiden. Bewusst wird der Unterschied, wenn die abgebildete Funktionalität der „GPSTMCAPI“ in ihrer Übertragung auf den „TMCProcessor“ nochmals verdeutlicht wird.

Die Funktionen „WM_GPSTMC_TMC_CHANGED“ und „WM_GPSTMC_TMC_-DELETE“ der „GPSTMCAPI“ können den Unterschied der beiden Tabellen erklären. Der File-Listener wird beim Eingang einer TMC-Meldung vom Typ „changed“ nach dem Erstellen des TMC-Objektes, die zu ersetzende TMC-Meldung über die Referenz des Location-Codes aus der Tabelle „tmc_active“ löschen, da diese verändert bzw. mit der aktuellen Meldung ersetzt wird. Beim Schreiben würden wieder beide Tabellen gefüllt werden. Dies führt dazu, dass in „tmc_oneday“ alle eingegangenen TMC-Meldungen gespeichert werden und in „tmc_active“ nur die aktuellen bzw. gültigen oder aktualisierten. Ferner wird nach 24 Stunden um 0 Uhr die aktuelle „tmc_oneday“ Tabelle in „tmc_oneday + Zeitstempel“ umbenannt und eine neue „tmc_oneday“ erstellt. Die „tmc_active“ Tabelle wird ebenfalls gelöscht und wieder neu erstellt. Als Resultat liegen in der Datenbank die Tabellen vom Zeitpunkt des ersten Starts des Dienstes für die vom Datenimporter gelieferten Daten, von jedem Tag in der entsprechen „tmc_oneday + Zeitstempel“ Tabelle vor.

Die Tabellen beinhalten alle Informationen, die „TMC_USERDATA“ liefert und dessen Interpretation, wie die folgende Tabelle zeigt.

Tabelle 9: Spalten der TMC-Tabellen

id serial NOT NULL	tmcTime VARCHAR(45)	rds VARCHAR (100)	wLoc2 VARCHAR (45)	lat VARCHAR(45)	lon VARCHAR (45)	wEvents INT(45)	wDecode VARCHAR (300)	wAge INT (45)	wSpeed_limit INT(45)
abySupplInfo VARCHAR (45)	bBidi bool(45)	bNew VARCHAR (45)	byCountryCode VARCHAR (45)	byDirection VARCHAR(45)	byDiversion VARCHAR (45)	byDuration VARCHAR (45)	byExtent VARCHAR (45)	byLtn VARCHAR(45)	byReserved1 VARCHAR (45)
byReserved2 VARCHAR (45)	bySide VARCHAR(45)	byTmcSource VARCHAR (45)	byUrgent VARCHAR (45)	cDetDiversions VARCHAR(45)	cQuantifiers VARCHAR (45)	explic_start VARCHAR (45)	explic_stop VARCHAR (45)	wLen VARCHAR(45)	frequency INT(45)
Type VARCHAR (45)									

Die Tabelle 9 zeigt die Spalten der TMC-Tabellen. Die orange gefärbten Zellen entsprechen der Übersetzung aus dem struct „TMC_USERDATA“ des Datenimporters und werden entsprechend zugeordnet hinterlegt. Die grün gefärbten Zellen entsprechen Zusatzinformationen oder die Decodierung der vom DLR geforderten Größen einer TMC-Meldung.

Der Erfassungszeitpunkt wird in der Spalte „tmcTime“ festgehalten, die Spalte „rds“ beinhaltet den gesamten RDS-String, so wie er vom Nachrichtensender geliefert wird. Die Spalten „lat“ und „lon“ stellen die Übersetzung des LocationCodes in WGS84-Koordinaten[42] dar. In der Spalte „frequency“ wird die vom Tuner eingestellte Frequenz gespeichert. Der „Type“ bezieht sich auf den „Type“ des Datenimporters bzw. auf den Typ der TMC-Meldung, der die Werte „New“, „Change“ und „Delete“ annehmen kann, siehe Abbildung 41 „TMCUserData“. Im folgendem Kapitel wird unter anderem diese Information benötigt, um die empfangen Daten zu analysieren.

[42]Das World Geodetic System (WGS-84) wurde 1984 als ein neues Geodaten-Referenzsystem als einheitliche Positionsangabe auf der Erde erdacht und beschreibt die Positionierung auf Basis eines Referenzellipsoiden.

Eine Besonderheit ist die Spalte „wDecode“. Die durch einen Delimiter[43] getrennten Informationen sollen später auf dem Web Frontend dargestellt werden, dazu mehr im Kapitel 8 „Darstellung der neuen “.

[43] Delimiter ist ein Trennzeichen oder Wort, das Sätze oder Wortgruppen teilt und wird in der Programmierung oft genutzt um Zeichenketten zu parsen.

6. TMC Datenanalyse

Für die TMC Datenanalyse wurde ein eigenes Modul geschrieben, dass die Darstellung der Daten mit der Java-API „JFreeChart“ nutzt. Diese Programmierschnittstelle ermöglicht dem Nutzer für die Auswertung der gesammelten TMC-Daten eine Darstellung in beliebigen Diagrammen jeglicher Form. Die Nutzung von anderen Analyseprogrammen wie „Matlab“ wäre ebenso möglich. Da zuvor auf die Java-Architektur zurückgegriffen wurde, können die erstellten Klassen zur Datenbankverbindung genutzt werden, um die vorhanden Daten für das Analyseprogramm bereitzustellen. In „Matlab“ wäre diese einfache Übertragung nicht möglich, da dieses Programm zwar eine JDBC-Schnittstelle anbietet, diese aber zur Registrierung ein Interface benötigt, das entsprechend noch zu programmieren wäre. [53] Im Folgenden soll eine quantitative und qualitative Auswertung der TMC-Meldung nur für Berlin vorgenommen werden. Diese Information ist wichtig, da nur Meldungen betrachtet werden die innerhalb der Breitengrade 52.28059 und 52.73945 und der Längengrade 12.88460 und 13.81158 liegen. Die Abbildung 53 zeigt diese Region.

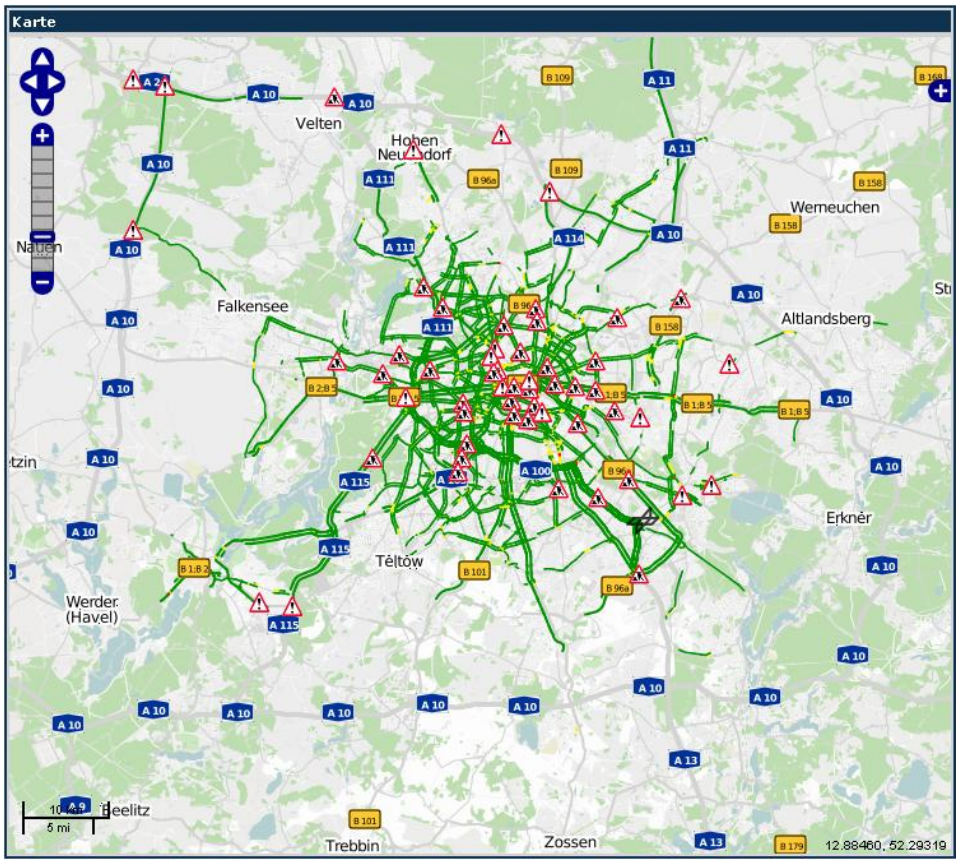


Abbildung 53: Region Berlin

Diese Eingrenzung ist nötig, da für die Qualitätsanalyse ein Abfahren der TMC-Meldungen nötig ist. Dies würde wiederum bei der Einbeziehung von TMC-Meldungen für das Ruhrgebiet, zu einer nicht akzeptablen Fahrzeit führen.

6.1. Quantitative Auswertung

Der Beginn der Datenanalyse soll klären, welche der Quellen aus Abschnitt 3.2.2. quantitativ die Beste ist. Hierzu soll auf Basis der gesammelten Daten, die Anzahl aller eingegangen TMC-Meldungen für einen Tag in Berlin bestimmt werden.

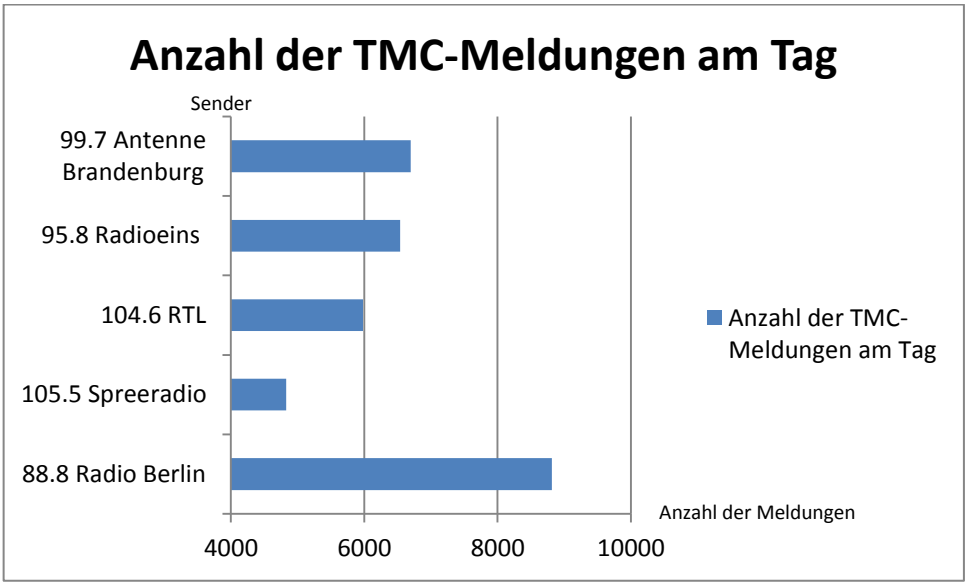


Diagramm 1: Anzahl der TMC-Meldung am Tag

Das Diagramm 1 „Anzahl der TMC-Meldung am Tag“ zeigt, wie häufig von den jeweiligen Radiosendern TMC-Meldungen empfangen wurden. Dabei ist ein exakter Vergleich in absoluten Zahlen problematisch, da die Daten nicht alle am selben Tag aufgenommen wurden. Dafür wären für jeden der 5 Radiosender 5 Empfangsgeräte nötig, die für diese Arbeit nicht vorhanden waren. Der quantitative Vergleich ist dennoch möglich, da dieses Diagramm ähnliche Tage zeigt. Jede der Zählungen fand an einem Montag statt. Aus jeweils 2 Montagen wurde ein Mittelwert für einen Sender gebildet, um tägliche Schwankungen ähnlicher Tage nicht zu stark in die Betrachtung mit einfließen zu lassen. Die folgende Tabelle 10 zeigt welche Tabellen bzw. welche Werte welcher Tage in das Diagramm einfließen.

Tabelle 10: Tage für Anzahlmessung

Radiosender	Tag 1	Tag 2
88.8 Radio Berlin	Tabelle vom: 15.08.2011	Tabelle vom: 19.09.2011
105.5 Spreeradio	Tabelle vom: 22.08.2011	Tabelle vom: 26.09.2011
104.6 RTL	Tabelle vom: 29.08.2011	Tabelle vom: 03.10.2011
95.8 Radioeins	Tabelle vom: 05.09.2011	Tabelle vom: 10.10.2011
99.7 Antenne Brandenburg	Tabelle vom: 12.09.2011	Tabelle vom: 17.10.2011

Um im Vorhinein sicherzustellen, dass nicht auf Grund gewisser Umstände, wie einem Papstbesuch mehr Verkehrsnachrichten gesendet werden, wurden folgende Anforderungen an die Tage gestellt.

Der TMC-Empfänger muss maximalen Empfang haben. Jeder der Tage muss ein Montag sein, kein Feier- und kein Ferientag. Ferner sollen keine Großveranstaltungen für diese Zeit in Berlin sein. Dies ist wichtig, wie die TMC-Tabelle für den 08.08.2011 (letzte Sommerferienwoche 2011) beispielsweise zeigt. An diesem Tag wurden über 10000 TMC-Meldungen von „Radio Berlin“ empfangen. Trotz der Bemühung relativ homogene Tage zu nutzen, wäre es falsch absolute Zahlen aus dem Diagramm ablesen zu wollen, da der TMC-Empfang bzw. dessen Aufkommen trotz der benannten Einschränkungen eine individuelle Tageskomponente, wie z.B. das Wetter, trägt.

So ist aus dem Diagramm, ohne auf konkrete Werte einzugehen, sehr gut eine Tendenz ersichtlich. Vom Sender „Radio Berlin“ wurden mehr TMC-Meldungen empfangen als von anderen Sendern in der Region Berlin. Ein Grund hierfür könnte seine exklusive Möglichkeit sein, direkt aus der Verkehrsmanagement Zentrale Berlin zu senden. [54]

Des Weiteren ist zu betrachten, ob der Radiosender „88.8 Radio Berlin“ wirklich mehr TMC-Meldungen sendet als andere. Die Möglichkeit besteht, dass dieser nur in kürzeren Intervallen

als andere Sender die Meldungen wiederholt und somit nicht mehr neue Meldungen sendet als beispielsweise „Spreeradio“. Hierzu folgendes Diagramm.

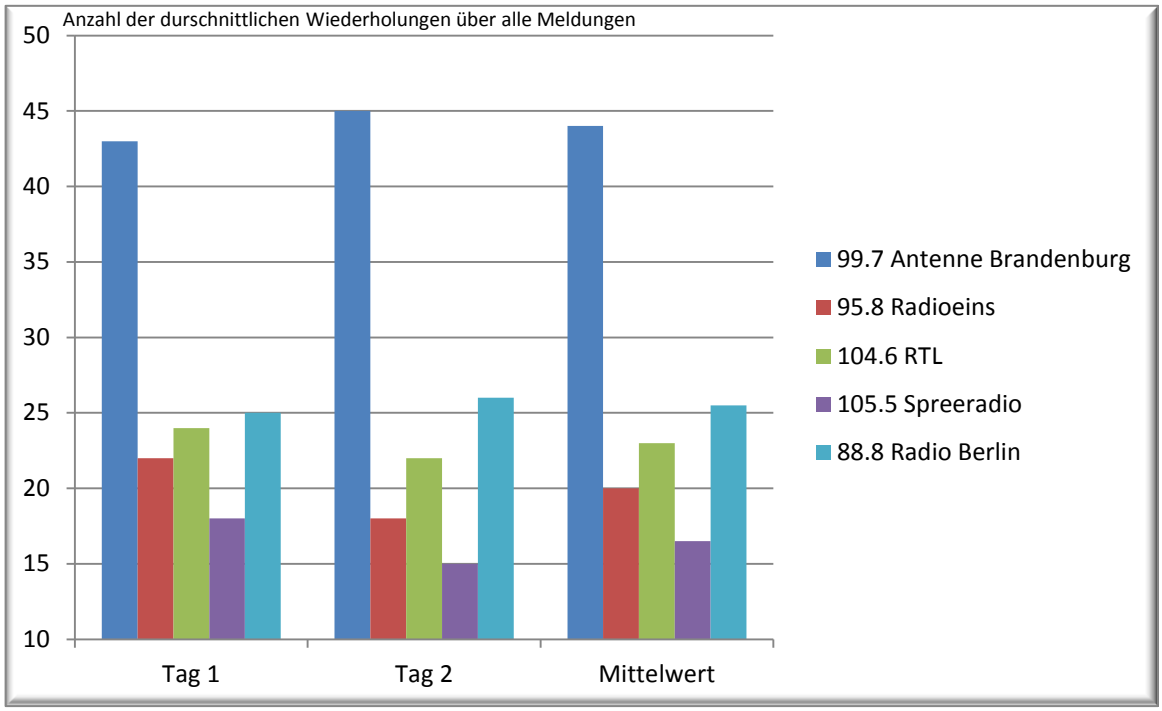


Diagramm 2: Wiederholungsrate von TMC-Meldungen

Das Diagramm 2 „Wiederholungsrate von TMC-Meldungen“ zeigt, wie oft im Mittel über alle TMC-Meldungen an einem Tag eine Meldung vom jeweiligen Sender wiederholt wird. Dabei werden die Werte relativ zu ihrer Übertragungsdauer betrachtet, da Verkehrsstörungen sich nacheinander im Tagesverlauf ereignen. Das Verfahren soll an einem Beispiel erklärt werden:

Um 8:50 Uhr am Montag, dem 15.08.2011 geschieht ein Unfall auf der A100. Dieses führt dazu, dass die Radiosender mit etwas Zeitverzug die TMC-Meldung senden. Konkret könnte der Radiosender „88.8 Radio Berlin“ ab 9:00 Uhr die entsprechende TMC-Meldung senden. In den nächsten 6 Stunden normalisiert sich der Verkehr an der Unfallstelle, da das Hindernis beseitigt wurde. Infolge dessen wird um 15:00 Uhr die Meldung das letzte Mal vom Sender ausgestrahlt. Wenn die Nachricht vom ersten Ausstrahlen bis zum letzten Ausstrahlen in sechs Stunden siebenmal gesendet wurde, entspricht das einer Wiederholungrate für den gesamten Tag, für eine Meldung, von vier mal sieben Wiederholungen also achtundzwanzig. Demnach bestimmt sich die Wiederholungrate einer TMC-Meldung (WrT) für den jeweiligen Tag mit:

$$\mathrm{WrT} = \frac{24}{\mathrm{h}} * WT$$

Die Variable h entspricht den Stunden in denen die TMC-Meldung empfangen wurde und WT ist die Anzahl der auftretenden Wiederholungen in diesem Zeitraum, für den zu betrachtenden Tag einer Meldung.

Der Wert (WrT) wird für jede erhaltende TMC-Meldung erstellt und anschließend werden die Teilergebnisse aufaddiert und durch die Anzahl der unterschiedlichen TMC-Meldungen geteilt.

$$\mathrm{MWr} = \frac{\sum WrT}{n}$$

Die so errechnete mittlere Wiederholungsrate (MWr) aller TMC-Meldungen eines Senders für diesen Tag wird im Diagramm 2 vermerkt. Entsprechend wird für den zweiten Tag verfahren. Die dritte Spalte „Mittelwert“ der X-Achse zeigt die Summe beider Werte für den jeweiligen Sender durch zwei geteilt. Allgemein wird dieser Wert bestimmt mit:

$$\mathrm{M} = \frac{\sum MWr}{n}$$

Der Mittelwert der mittleren Wiederholungsrate (M) ist die Summe aller MWr durch deren Anzahl (n). So werden die Werte für die beiden Tage gegenübergestellt und zusammengefasst. Für diese Analyse wurden die jeweiligen Datenbanktabellen für die bereits erwähnten Montage genutzt.

Im Diagramm 2 ist auffällig, dass der Radiosender „Antenne Brandenburg“, eine relativ hohe Wiederholungsrate besitzt. Zusammen mit den Aussagen aus dem Diagramm 1 ergibt sich für die Sender folgende Betrachtung:

Der Radiosender „Antenne Brandenburg“ sendet im Vergleich zu den anderen Sendern die zweit meisten TMC-Meldungen mit im Mittel etwas weniger als 6700 TMC-Meldungen für die betrachteten Montage, wobei jede Meldung im Tagedurschnitt 44 mal gesendet wird, siehe Diagramm 2. Effektiv werden von „Antenne Brandenburg“ somit 152 TMC-Meldungen an einem solchen Montag herausgegeben. Für einen sinnvollen Vergleich, sollen die Effektivwerte der anderen Sender bestimmt werden.

Zur Bestimmung des Effektivwertes wird folgende Formel benötigt:

$$\mathrm{E} = \frac{A}{\mathrm{M}}$$

E = Effektivwert, A = gesamte Anzahl der Meldungen, M = mittlere Wiederholungsrate

Diese Formel wird auf alle Radiosender angewendet und führt zu der folgenden Tabelle.

Tabelle 11: Effektivwerte der Sender

Radiosender	Anzahl der Meldungen	Mittlere Wiederholungsrate	Effektivwert (gerundet)
88.8 Radio Berlin	8815	25.5	346
105.5 Spreeradio	4831	16.5	293
104.6 RTL	5982	23	260
95.8 Radioeins	6541	20	327
99.7 Antenne Brandenburg	6696	44	152

Wie aus der Tabelle 11 ersichtlich, ist der Radiosender „88.8 Radio Berlin“ nicht nur in Absolutzahlen der beste Sender, sondern auch der Sender der tatsächlich am meisten unterschiedliche TMC-Meldungen herausgibt. Der Radiosender „Antenne Brandenburg“ stellt sich als der Sender heraus, der am wenigsten neue Meldungen sendet und somit das Verkehrsgeschehen am schlechtesten abbildet. Selbst saisonale Schwankungen innerhalb der analysierten Tage, können den Unterschied von mehr als 100 Prozent zwischen dem schlechtesten und dem besten Sender nicht erklären. Im Vergleich wurden an einem Ferientag (Tabelle vom 08.08.2011) von „Radio Berlin“ 10056 TMC-Meldungen empfangen. Gegenüber dem betrachteten Montag aus Tabelle 11 mit 8815 empfangen Meldungen beträgt der Unterschied 1241 Nachrichten. Bei einer mittleren Wiederholungsrate von 25.5 sind das rund 49 neue TMC-Meldungen für diesen Tag. Bei einem Abstand von mehr als 190

effektiven Meldungen zwischen „Antenne Brandenburg“ und „Radio Berlin“ würde demnach selbst eine größere Inhomogenität der Tage zueinander die gleiche Tendenz zeigen. In Folge dieser Analyse soll der Radiosender „88.8 Radio Berlin“ für die weiteren Betrachtungen herangezogen werden.

6.2. Qualitative Auswertung

Die Qualität der TMC-Meldungen zu beurteilen ist schwierig, da diese vom Radiosender abhängig ist. Ein wichtiges Qualitätsmerkmal ist der Wahrheitsgehalt der TMC-Meldungen, dessen Bestimmung für den urbanen Bereich Berlin in keiner der genutzten Literaturen zu finden ist. [45] [34] [35] [36] [8] Ein Grund dafür ist, dass die Bestimmung der Fehlerrate nicht nur gebietsbezogen, sondern auch vom Sender abhängig ist. Dies ist der Tatsache geschuldet, dass jeder Radiosender über die Herausgabe seiner TMC-Meldungen selber bestimmt. Somit soll in diesem Abschnitt eine Fehlerrate der TMC-Meldungen für die zuvor genannten und analysierten Sender „88.8 Radio Berlin“, „105.5 Spreeradio“, „104.6 RTL“, „95.8 Radioeins“ und „99.7 Antenne Brandenburg“ erhoben werden. Für diese Bestimmung wurde eine Messkampagne des DLR vom 26.09.2011 bis 30.09.2011, um die Analyse von Baustellen erweitert. Diese Kampagne dient eigentlich der Qualitätsbestimmung von FCD, bei dem 5 Messfahrtzeuge im gesamten urbanen Bereich Berlin unterwegs sind und von 7 bis 17 Uhr Daten erheben. Die genaue Beschreibung der Messfahrten ist für die weitere TMC-Analyse nicht relevant und soll im Folgendem nicht weiter beschrieben werden. Wichtig ist, dass für die Bestimmung der Fehlerrate der gesendeten TMC-Meldungen, das Fahrprotokoll um die Spalte Baustellen erweitert wurde. Die Aufgabe der Durchführenden war es, beim Erkennen einer Baustelle den Zeitpunkt in das Protokoll einzutragen. Mit Hilfe der gesammelten Datensätze ist es anschließend möglich, den Ort der Baustelle zu bestimmen. Die Abbildung 54 zeigt die erhobenen Daten vom 26.09.2011 von einem Fahrzeug in der KML-Darstellung von Google-Earth. Jeder Datensatz ist mit einem roten X markiert und enthält einen Zeitstempel und die gefahrene Geschwindigkeit. Das in der Abbildung geöffnete Popupfenster ermöglicht, den Inhalt des spezifischen Datensatzes zu lesen. Dieser bezieht sich auf den Zeitpunkt 7 Stunden 16 Minuten und 49 Sekunden des 26.09.2011, bei dessen Aufnahme rund 17km/h gefahren wurden.

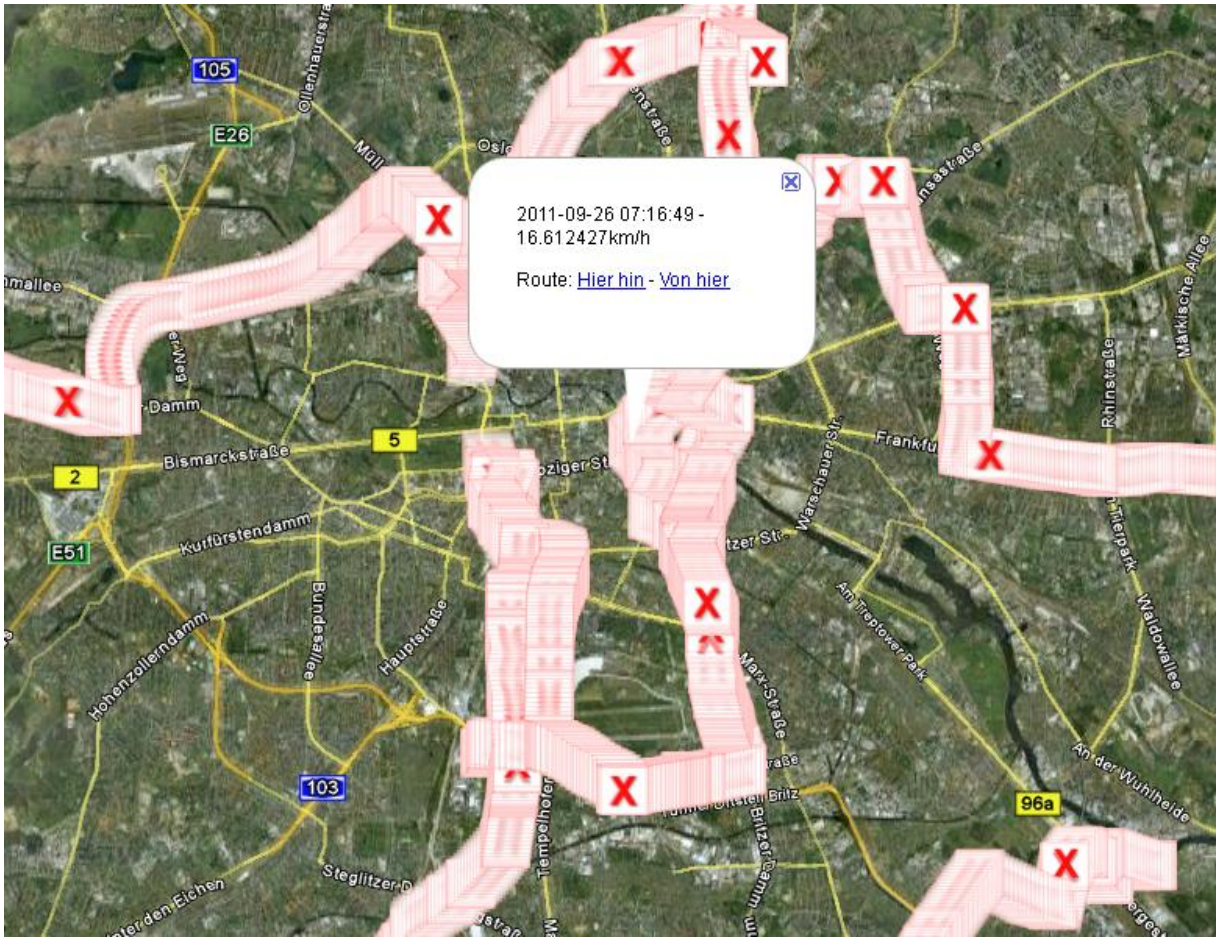


Abbildung 54: Messkampagne FCD

Anschließend müssen aus der Datei alle Daten gefiltert werden, die laut Protokoll Baustellen sein sollen. Die gefundenen Baustellen eines Fahrzeuges werden mit den TMC-Baustellen verglichen, die in Abbildung 55 für den 26.09.2011 gezeigt werden.

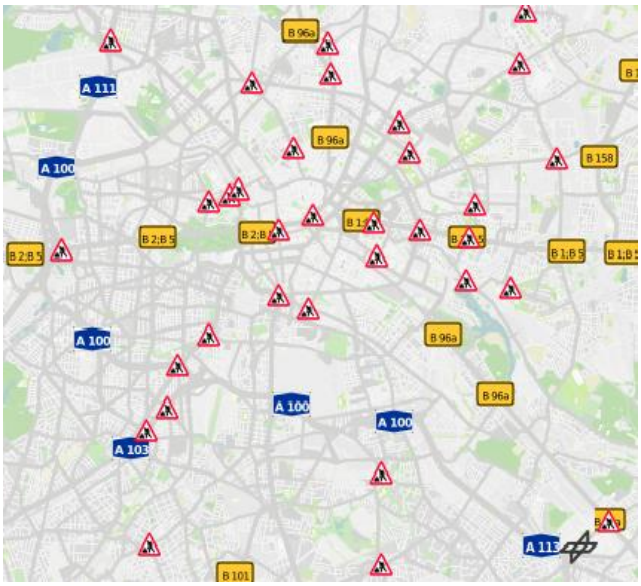


Abbildung 55: Baustellen TMC

Wenn eine Baustelle aus der Messfahrt mit einer aus den TMC-Daten übereinstimmt, dann werden die gefilterten Baustellen in der KML-Datei (grünes B) in ein anderes Symbol geändert (rotes B) und der zugehörige Schriftzug wird von „nicht in TMC“ in „Baustelle“ geändert. Des Weiteren werden alle verbleibenden TMC-Baustellen auf der Route, die nicht von der Messkampagne erfasst wurden, mit einem hellblauen Icon markiert und mit der Beschriftung „nicht in Messkampagne“ versehen.

Zusammenfassend zeigen alle Icons mit einem B die Baustellen nach Messkampagne an, wobei ein rotes B eine Baustelle kennzeichnet, die auch in den TMC-Meldungen vorhanden ist. Des Weiteren zeigt ein hellblaues Icon die TMC-Baustellen, die nicht in der Messkampagne zu finden sind. Die Abbildung 56 fasst das erstellte Ergebnis für das erste Fahrzeug für den 26.09.2011 in der erstellten KML-Datei zusammen.

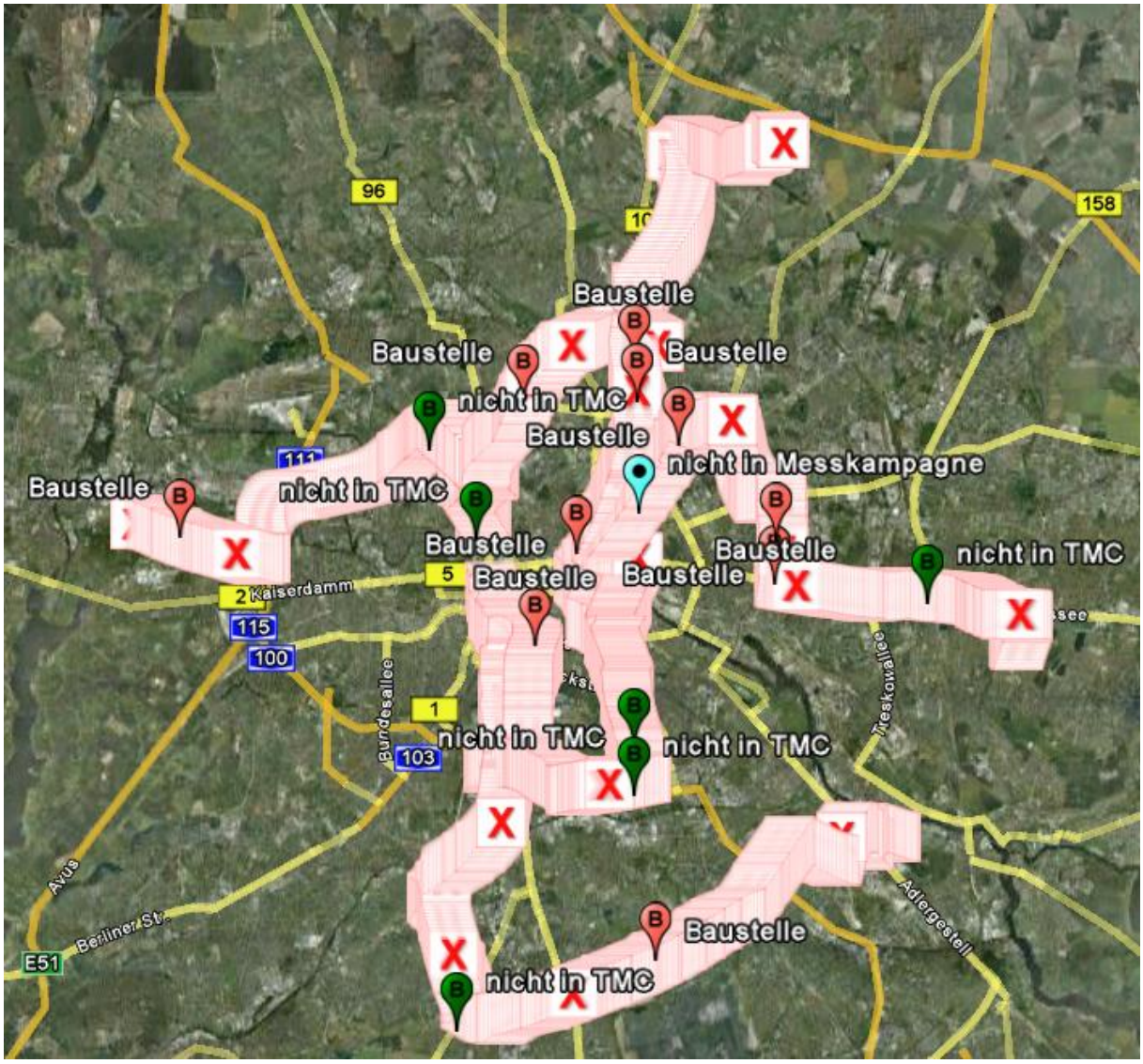


Abbildung 56: Messkampagne Abgleich

Aus der Abbildung 56 kann nun folgender Informationsgehalt abgeleitet werden. Die roten und hellblauen Icons sind in der Summe 11, demnach zeigen die TMC-Daten für den 26.10.2011 11 Baustellen für die befahrene Route, von denen 10 nach der Messkampagne tatsächlich vorhanden sind. Die hellblaue Baustelle Prenzlauer Berg, Ecke B2 ist nicht real, was einem Fehler von rund 9 Prozent entspricht. Ferner ist erkennbar, dass insgesamt 16 Baustellen von dem Messfahrzeug detektiert wurden (alle B‘s), von denen 10 vom Radiosender „Antenne Brandenburg“ gesendet wurden (rote B‘s). Dies entspricht einer Abdeckung von 62,50 Prozent. Zusammengefasst beschreiben die Daten den Radiosender beim Senden der TMC-Baustellen mit einem Fehler von 9 Prozent, wobei 62,50 Prozent der tatsächlich existierenden Baustellen übertragen wurden.

Diese Analyse zeigt das angewendete Verfahren für ein Fahrzeug, des Radiosenders „Antenne Brandenburg“. Für alle 5 Fahrzeuge soll im Folgenden die Ergebnisse präsentiert werden, bei denen jeden Tag ein anderer Sender durch das oben stehende Verfahren validiert wird.

Tabelle 12: Vergleichstabelle Messkampagne

Radiosender (Datum)	Fahrzeug 1	Fahrzeug 2	Fahrzeug 3	Fahrzeug 4	Fahrzeug 5
99.7 Antenne Brandenburg (26.09.2011)	16 Baustellen 11 in TMC 10 gleich	22 Baustellen 15 in TMC 15 gleich	8 Baustellen 8 in TMC 8 gleich	14 Baustellen 8 in TMC 5 gleich	26 Baustellen 18 in TMC 16 gleich
88.8 Radio Berlin (27.09.2011)	16 Baustellen 10 in TMC 10 gleich	22 Baustellen 15 in TMC 15 gleich	8 Baustellen 8 in TMC 8 gleich	14 Baustellen 8 in TMC 8 gleich	26 Baustellen 18 in TMC 17 gleich
104.6 RTL (28.09.2011)	16 Baustellen 9 in TMC 8 gleich	22 Baustellen 10 in TMC 9 gleich	8 Baustellen 8 in TMC 8 gleich	14 Baustellen 8 in TMC 8 gleich	26 Baustellen 15 in TMC 15 gleich
95.8 Radioeins (29.09.2011)	16 Baustellen 10 in TMC 10 gleich	22 Baustellen 11 in TMC 11 gleich	8 Baustellen 8 in TMC 8 gleich	14 Baustellen 8 in TMC 5 gleich	26 Baustellen 10 in TMC 9 gleich
105.5 Spreeradio (30.09.2011)	16 Baustellen 5 in TMC 5 gleich	22 Baustellen 11 in TMC 11 gleich	8 Baustellen 8 in TMC 8 gleich	14 Baustellen 5 in TMC 5 gleich	26 Baustellen 10 in TMC 10 gleich

Die Tabelle 12 zeigt die Radiosender im Zusammenhang mit den erstellten Daten. Dabei wurden alle TMC-Baustellen für einen Radiosender an einem Tag der Messkampagne für den gleichen Tag gegenübergestellt. Abzulesen ist, dass die Fahrzeuge innerhalb der gesamten Woche immer die gleiche Anzahl von Baustellen erfasst haben, da die Zahl der Baustellen für ein Fahrzeug für jeden Tag der Woche gleich ist. Des Weitern zeigt die erste Zeile, dass am Montag dem 26. Oktober das Fahrzeug 2 auf seiner Route 22 Baustellen gefunden hat, von denen 15 in der TMC-Datenbank vorhanden sind. Alle 15 Baustellen sind auf die Daten der Messkampagne übertragbar. Demnach ergibt sich für Fahrzeug 2 eine Abdeckung von TMC-Baustellen zu den tatsächlichen Baustellen von rund 68 Prozent, wobei diese 68 Prozent keinen Fehler haben, da alle Baustellen nach der Messkampagne existieren. Bei der Auswertung aller Tage für alle Fahrzeuge ergibt sich die Tabelle 13.

Tabelle 13: Messkampagne Vergleich in Prozent einzeln

Radiosender (Datum)	Fahrzeug 1 Abdeckung % Richtig %	Fahrzeug 2 Abdeckung % Richtig %	Fahrzeug 3 Abdeckung % Richtig %	Fahrzeug 4 Abdeckung % Richtig %	Fahrzeug 5 Abdeckung % Richtig %
99.7 Antenne Brandenburg (26.09.2011)	62,50 90,91	68,18 100,00	100,00 100,00	35,71 62,50	61,54 88,89
88.8 Radio Berlin (27.09.2011)	62,50 100,00	68,18 100,00	100,00 100,00	57,14 100,00	65,38 94,44
104.6 RTL (28.09.2011)	50,00 88,89	40,91 90,00	100,00 100,00	57,14 100,00	57,69 100,00
95.8 Radioeins (29.09.2011)	62,50 100,00	50,00 100,00	100,00 100,00	35,71 62,50	34,62 90,00
105.5 Spreeradio (30.09.2011)	31,25 100,00	50,00 100,00	100,00 100,00	35,71 100,00	38,46 100,00

Die Tabelle 13 zeigt die nach der zweiten Kommastelle gerundeten Werte für die Abdeckung und dessen Richtigkeit in Bezug von Messkampagne und TMC-Baustellen nach den folgenden Formeln für Abdeckung und Richtigkeit:

$$\text{Abdeckung} = \frac{100}{\text{Baustellen}} * gleich$$

$$\text{Richtigkeit} = \frac{100}{\text{in TMC}} * gleich$$

Die Bezeichner in den Formeln beziehen sich direkt auf die Ausdrücke in den Tabellen 12 und 13.

Bei der Betrachtung der Tabelle 13 ist auffällig, dass die Route von Fahrzeug 3 nach Fahrprotokoll genauso viele echte Baustellen enthält wie TMC-Baustellen und dass alle Meldungen im System für jeden Tag und jeden Sender übereinstimmen. Bei der Betrachtung der Routen zueinander, konnten keine signifikanten Unterschiede zwischen den Routen festgestellt werden, wie z.B. dass das Fahrzeug 3 nur auf Bundesstraßen oder nur auf Nebenstraßen fuhr. Demnach könnte die geringe Anzahl von Baustellen dazu führen, dass jeder Sender alle Baustellen dieser Route richtig sendete. Dies würde aber nicht erklären, warum keine falschen TMC-Baustellen für die Route gesendet wurden. Somit ist in Betracht zu ziehen, dass es keinen kausalen Zusammenhang für diese Daten gibt. Demzufolge ist es ein Zufall, dass für diese eine Woche alle Radiosender diese Baustellen, auf dieser Route richtig sendeten. Die weitere Analyse bedarf einer weiteren Messkampagne, die die Route des Fahrzeuges 3 wieder aufnimmt und somit die Möglichkeit gibt, die Messwerte zu bestätigen oder als eine Zufallsmomentaufnahme zu kennzeichnen.

Bevor eine Tendenz zum TMC-Sendeverhalten der Radiosender getroffen werden kann, soll eine Zusammenfassung der erstellten Prozente über alle Fahrzeuge ein allgemeineres Bild der jeweiligen Abdeckung und Gleichheit bzw. Fehlerrate geben.

Tabelle 14: Messkampagne Vergleich in Prozent

Radiosender (Datum)	Mittelwert aller Fahrzeuge Abdeckung % Richtig %	Mittlerer Fehler Fehlerrate %
99.7 Antenne Brandenburg (26.09.2011)	65,59 88,46	11,54
88.8 Radio Berlin (27.09.2011)	70,64 98,89	1,11
104.6 RTL (28.09.2011)	61,15 95,78	4,22
95.8 Radioeins (29.09.2011)	56,57 90,50	9,50
105.5 Spreeradio (30.09.2011)	51,09 100,00	0,00

Die Tabelle 14 zeigt die Mittelwerte für jeden Radiosender aus den Daten der 5 Fahrzeuge. Um das Ergebnis übersichtlicher zu gestalten soll ein Diagramm die Auswertung vereinfachen.

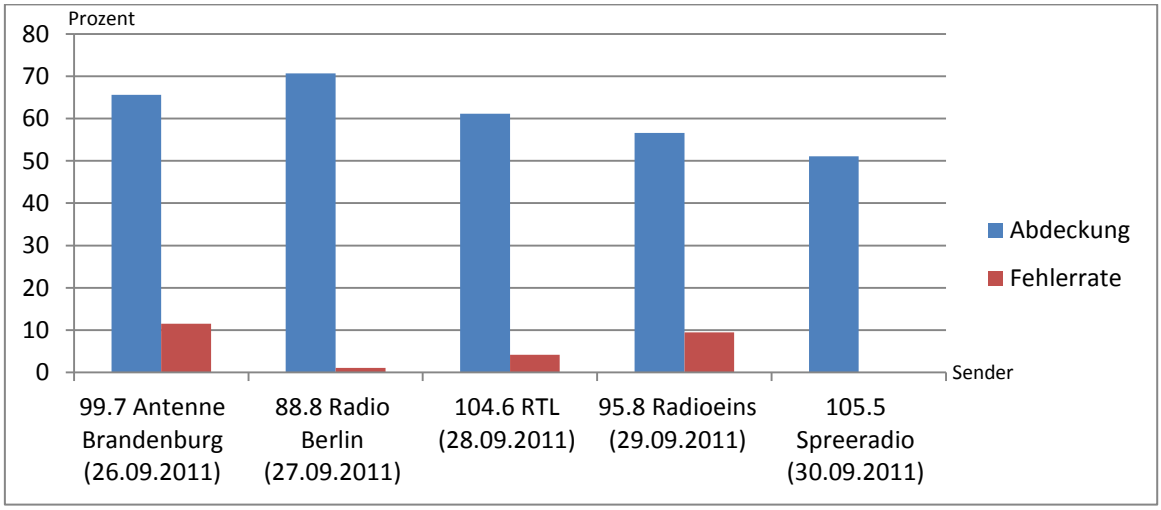


Diagramm 3: Messkampagne Vergleich in Prozent

Das Diagramm 3 zeigt die Tabelle 14 in einem Balkendiagramm, bei dem die Prozente über den Radiosender angezeigt werden. Der blaue Balken gibt die Abdeckung an, also wie viele TMC-Baustellen im Vergleich zu den realen Baustellen vom jeweiligen Radiosender gesendet werden. Der rote Balken gibt an, wie viele der gesendeten TMC-Baustellen falsch sind und in der Realität nicht existieren.

Bei der Auswertung der Qualität der einzelnen Sender wird nun ersichtlich, dass der Radiosender „Spreeradio“ am 30.09.2011 nur richtige TMC-Baustelle herausgegeben hat, die Fehlerrate ist 0. Der Sender mit der höchsten Fehlerrate ist „Antenne Brandenburg“ mit rund 12 Prozent, wobei dessen Abdeckung den zweitbesten Wert erreicht hat. Bei der Betrachtung der 3 ausstehenden Sender zeigt sich, dass „Radio Berlin“ mit der höchsten Abdeckung, von rund 71 Prozent die zweitniedrigste Fehlerrate von rund 1 Prozent besitzt. Der Drittbeste in beiden zu bestimmenden Werten ist der Radiosender „104.6 RTL“. Der Sender „Radioeins“ zeigt die zweitschlechteste Abdeckung von rund 57 Prozent mit der zweithöchsten Fehlerrate von rund 10 Prozent. Somit zeigt dieser Sender zusammengefasst die schlechteste Qualität dieser Auswertung. Bei der Wahl eines Radiosenders zum TMC-Empfang sollte einer der beiden Radiosender „Radio Berlin“ oder „Spreeradio“ als Quelle genutzt werden. Dabei sollte anhand der Betrachtung von Fehlerrate und Abdeckung je nach Qualitätswert sondiert werden. Wenn eine hohe Abdeckung von TMC-Meldungen, mit einer möglichst geringen Fehlerrate benötigt wird, sollte der Radiosender „Radio Berlin“ genutzt werden. Ist die Abdeckung relativ egal und der Hauptschwerpunkt liegt auf einer möglichste geringen Fehlerrate, dann sollte der Radiosender „Spreeradio“ im urbanen Bereich Berlin als TMC-Quelle genutzt werden.

Zu berücksichtigen ist bei dieser Analyse, dass jeder Radiosender nur auf Basis eines Tages bewertet wurde und lediglich durch 5 Fahrzeuge validiert werden konnte. Daraus folgt, dass die Fehlerrate vom Radiosender „Spreeradio“ in der Realität unter Umständen höher liegen wird und die Fehlerraten anderer Sender vielleicht niedriger. Des Weiteren wäre eine genauere Analyse der TMC-Meldungen möglich gewesen, wenn das gesamte TMC-Spektrum der Meldungen hätte erfasst werden können bzw. dürfen, wie Staus, Fahrbahneinengungen usw.. Da der Schwerpunkt der FCD-Messkampagne aber nicht auf TMC lag, konnten nur Baustelleninformationen gesammelt und in Folge dessen zur Analyse herangezogen werden. Der Prozentsatz der Baustellenmeldungen ist zwar sehr groß, wie aus dem Abschnitt 6.4. bzw. dem enthaltenden Diagramm 4 abzulesen ist, dennoch ist aus der Analyse lediglich eine Tendenz abzulesen. Zur Bestimmung von aussagekräftigen Werten sollte eine TMC-Messkampagne über einen Zeitraum von mindesten 10 Wochen gestartet werden, bei der

jeden Tag jede TMC-Meldungen vor Ort überprüft wird. Nur eine solche Qualitätsbestimmung kann wissenschaftlich verwertbare Daten liefern.

Die vorgestellte Analyse kann dennoch eine Tendenz für die Fehlerrate der Radiosender vermitteln und soll als Sensibilisierung beim Umgang mit den vorliegenden TMC-Daten dienen. Klar geht aus der Analyse hervor, dass die Fehlerrate von TMC abhängig vom Sender ist und diese sich von Sender zu Sender unterscheiden wird. Bei der Betrachtung von TMC-Meldungen sollte davon ausgegangen werden, dass nicht alle Meldung real existieren.

6.3. Positionsauswertung

Bei der Betrachtung der Positionen der eingehenden TMC-Meldungen vom Radiosender „88.8 Radio Berlin“ ist zu sehen, dass nicht nur Meldungen für Berlin erstellt werden, sondern für gesamt Deutschland.

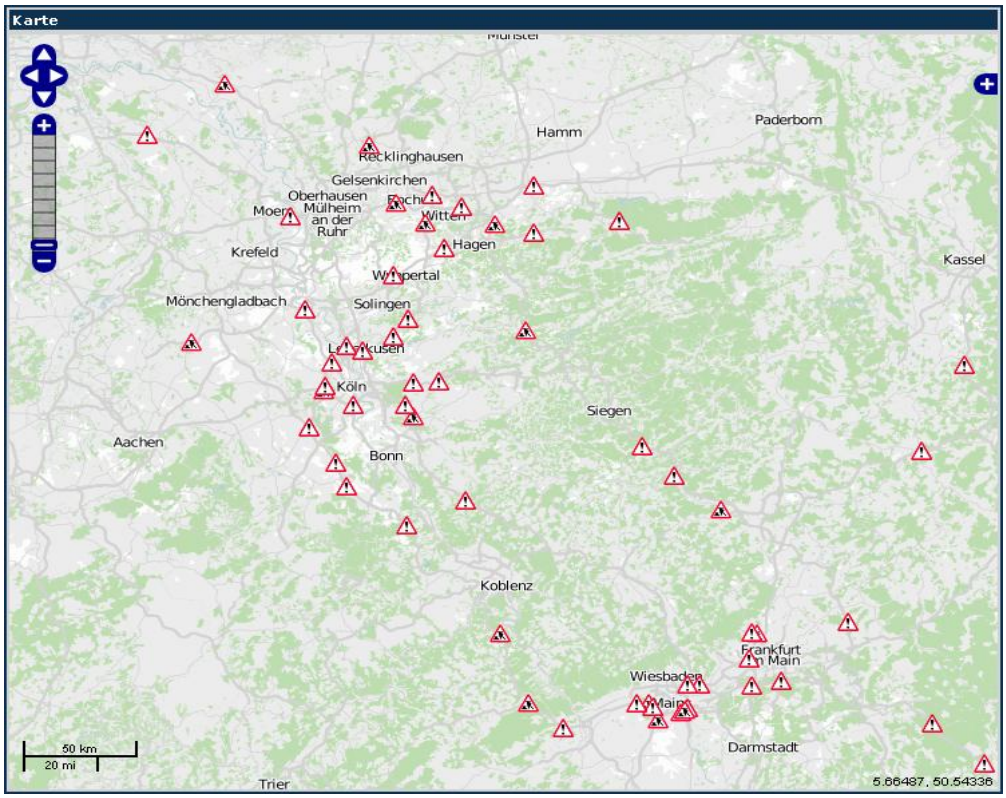


Abbildung 57: TMC Positionen

Die Abbildung 57 zeigt einen Ausschnitt aus der Anzeige des Cityrouters. Das hier betrachtete Ruhrgebiet soll stellvertretend für alle Regionen in Deutschland zeigen, dass die aus Abschnitt 6.1. beschriebenen Radiosender, überregionale TMC-Meldungen im gesamten Bundesbereich vergeben. Diese Meldungen werden innerhalb der Arbeit nicht weiter

betrachtet, da eine neue Übersicht der Verkehrslage für den urbanen Bereich Berlin und nicht für gesamt Deutschland erstellt werden soll.

6.4. Ereignisauswertung

Die TMC-Daten sind ereignisbezogene Informationen, die für eine bestimmte Position erstellt werden. Ausgehend von dieser Information können Regionen zusammengefasst werden, um eine übergeordnete Meldung zu generieren. Dies soll im Folgenden an einem Beispiel erklärt werden.

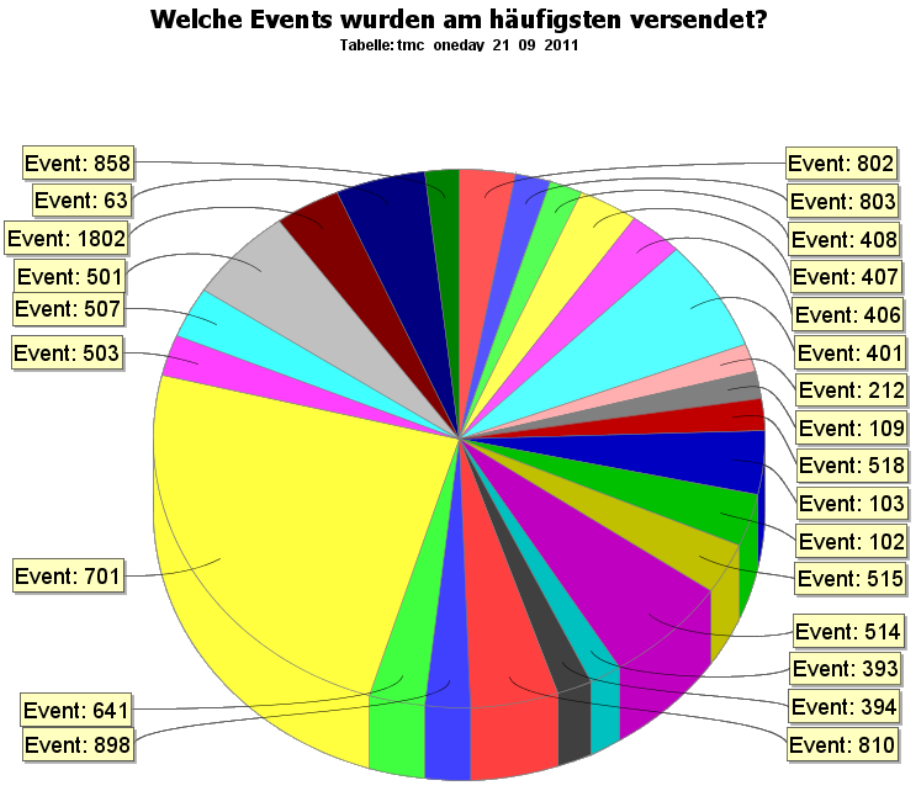


Diagramm 4: Ereignisübersicht Berlin

Das Diagramm 4 zeigt die 25 häufigsten Ereignisse im Bereich Berlin. Um in die Analyse aufgenommen zu werden, muss eine TMC-Meldung innerhalb der Koordinaten aus Abbildung 53 liegen. Auffällig ist, dass das Ereignis mit dem Event-Code 701 sehr häufig an diesem 21.09.2011 aufgetreten ist. Dieser Event-Code steht für Baustelle, demnach ist im Bereich Berlin vermehrt mit Beeinträchtigungen durch Baustellen zu rechnen. Darauf folgen die Ereignisse 514 (Fahrbahnverengung) und 401 (Sperrung), was als eine direkte Folge des hohen Baustellenaufkommens zu interpretieren ist. Dieses Verfahren lässt sich für beliebige Regionen übertragen und eingrenzen. Ferner ist es möglich, ganze Straßen aus der Datenbank

positionsspezifisch zu filtern und entsprechende Meldungen zu generieren. Dazu mehr in einem Ausblick in Kapitel 12. Momentan erstellt die „TMC Process Unit“ nur für den gesamten Bereich Berlin eine übergeordnete Meldung, die bei Bedarf dem Cityrouter hinzugefügt werden kann.

6.5. Vergleich FCD & TMC

Die hier getroffenen Aussagen basieren auf den Beobachtungen der Gegenüberstellung der folgenden Straßenabschnitte: Friedrichstraße, Ecke Leipziger bis Friedrichstraße, Ecke Torstraße, Adlergestell Auffahrt Seegraben bis Adlergestell, Ecke Köpernicker Straße und Buschkrugallee bis Dreieck Charlottenburg. Da die gezeigten Beobachtungen für jede der Strecken zutreffen, soll letztere stellvertretend zur Beschreibung herangezogen werden.

Der Vergleich der FCD- und TMC-Daten ist nur in ihrem Bezug zur Geschwindigkeit möglich, da FCD und TMC sonst keine weiteren gemeinsamen verwertbaren Werte besitzen. Ferner ist zu bedenken, dass TMC vor allem Ereignisse zu einer bestimmten Position liefert und die Geschwindigkeit eine Zusatzinformation ist, die nicht angegeben werden muss. Demnach mussten Strecken gewählt werden, die möglichst viele TMC-Meldung mit einem Geschwindigkeitsbezug enthalten. Ein Repräsentant mit den beschriebenen Charakteristika ist die Stadtautobahn A100. Diese Straße wird zum einen von vielen Taxis befahren und zum anderen liegt auf Grund der Wichtigkeit der Straße für den Berliner Verkehr ein gesteigertes Interesse bei den Radiosendern vor, diesen Abschnitt möglichst oft mit Zusatzinformation (Geschwindigkeit) zu versorgen. Der genau betrachtete Abschnitt beginnt Buschkrugallee und endet Dreieck Charlottenburg. Die Abbildung 58 zeigt diese Strecke mit aneinander gesetzten blauen Dreiecken über der gesamten Strecke.

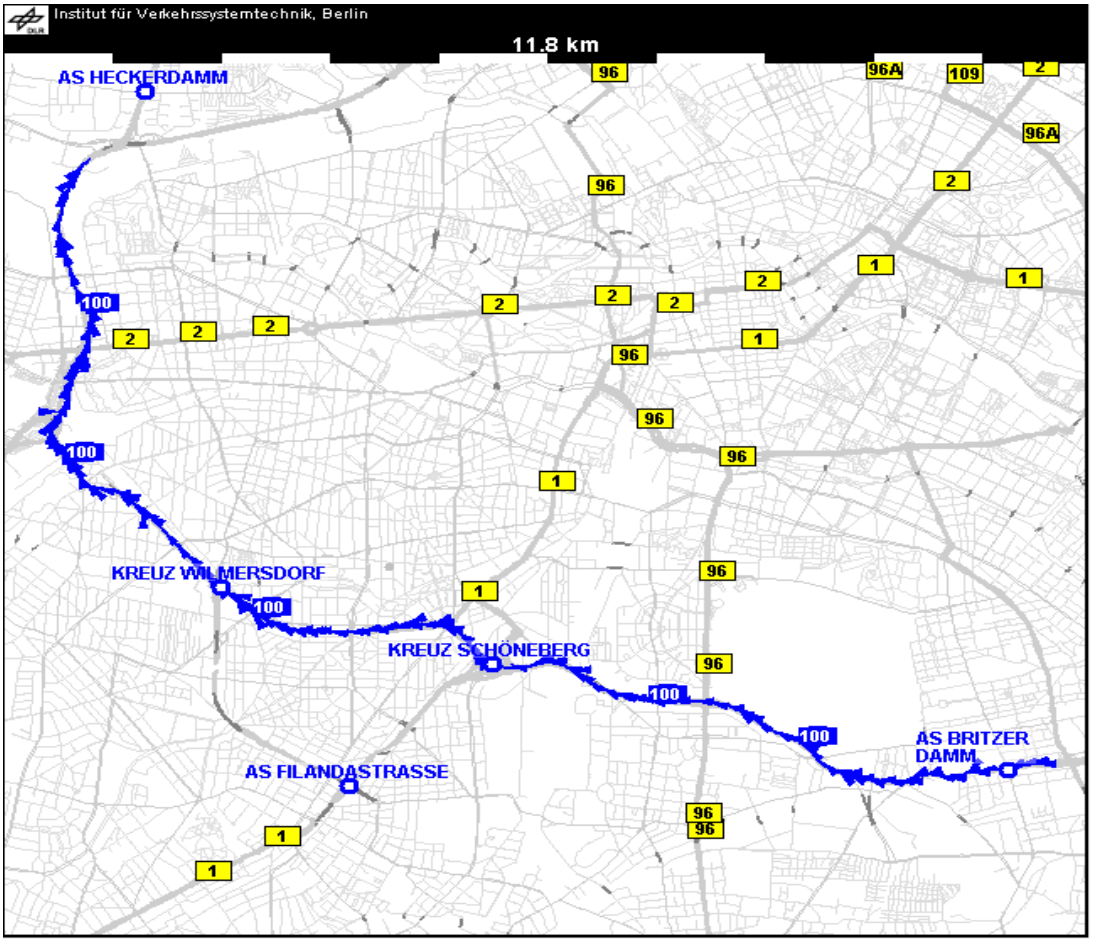


Abbildung 58: FCD & TMC A100

Für diese Strecke wird ein Diagramm erstellt, das die gefahrene Geschwindigkeit in Kilometer pro Stunde über der Tageszeit in Stunden anzeigt. Die Abbildung 59 vereint die TMC- und FCD-Meldungen für den 27.09.2011. Zusehen ist zum einen die Anzahl der Fahrzeuge, die für die Bestimmung der FC-Daten genutzt wurden (grüne Dreiecke), diese haben ihren Bezug zur rechten Achse. Des Weiteren zeigen die gelben Vierecke die entsprechend gefahrene Geschwindigkeit. Die rote Linie zeigt den gefitteten[44] Wert für die erfassten Geschwindigkeiten, dieser Wert entspricht vereinfacht dem Mittelwert. Die blauen Punkte sind die TMC-Meldungen mit Geschwindigkeitsbezug für diesen Tag.

[44] Gefittete Werte sind erstellte Daten, die mittels Algorithmen bestimmt werden, welche wiederum zum Teil Werte genutzt haben, die aus anderen erstellt wurden.

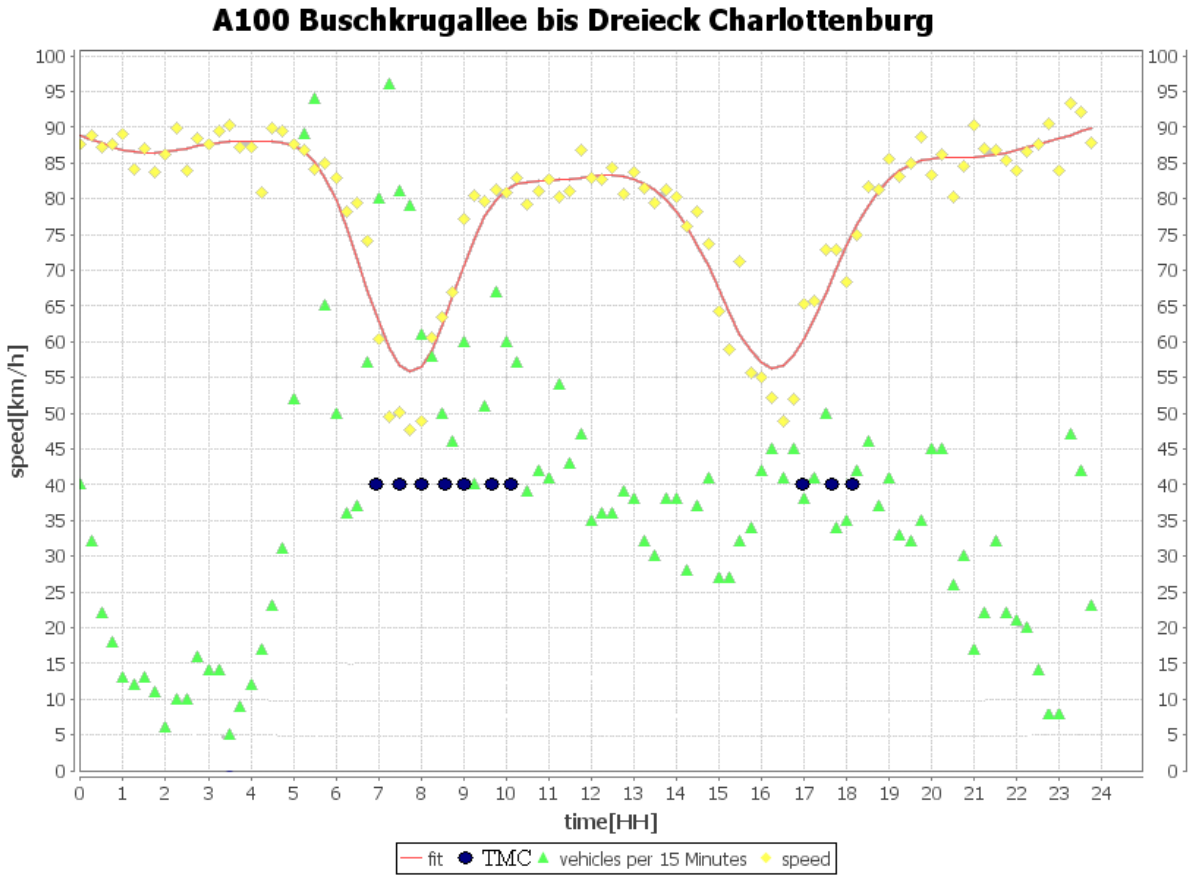


Abbildung 59: TMC & FCD Vergleich

Wie die Abbildung 59 zeigt, wurden für diesen Streckenabschnitt zehn TMC-Meldungen mit der Zusatzinformation Geschwindigkeit vom Radiosender „88.8 Radio Berlin“ für diesen Tag herausgegeben. Diese decken sich grob mit den Zeitpunkten der Verkehrsstörungen, die auch im FCD-System erfasst wurden. Auffällig ist, dass nach FCD die Geschwindigkeit von 7 bis 10 Uhr nie unter 55 km/h fiel. Die herausgegebene Zusatzinformation bei TMC zeigt für diesen Zeitraum eine Geschwindigkeit von 40 km/h an. Ein ganz ähnliches Phänomen tritt zwischen 16 und 18 Uhr auf. Beide Systeme zeigen eine Beeinträchtigung der Geschwindigkeit, aber nicht im gleichen Maße. Ferner ist zu erkennen, dass die TMC-Meldungen länger eine Beeinträchtigung zeigen als FCD. So wurde nach FCD um 9 Uhr schon über 70 km/h gefahren, TMC hingegen zeigt bis 10 Uhr eine Geschwindigkeit von 40 km/h an. Ebenso ist der Verzug der TMC-Meldungen am zweiten Abschnitt von 16 bis 18 Uhr gut zu erkennen. Um 17 Uhr ging die erste TMC-Meldung mit der Zusatzinformation 40 km/h ein, wo hingegen im FCD-System seit einer Stunde ein Geschwindigkeitsabfall auf dieser Strecke gemessen wurde. Diese Abweichungen lassen sich mit den aus Kapitel 3. beschriebenen Grundlagen erklären. Das FCD-System erfasst eine Verkehrsstörung in dem Moment, wenn ein Taxi den entsprechenden Abschnitt befährt. TMC enthält eine entsprechende Störung erst, wenn z.B. die Polizei an dem Unfallort eintrifft oder Radiohörer

bei dem Sender anrufen, um eine entsprechende Störung zu melden. Zu diesem Zeitpunkt können schon mehrere Taxis die Beeinträchtigung passiert haben. Dieser Erfassungsunterschied wird sehr gut deutlich, wenn man sich die beiden Abschnitte aus Abbildung 59 noch einmal genau ansieht. Ab 6 Uhr zeigt sich im FCD-System eine Abnahme der Geschwindigkeit. Gegen 7 Uhr ist beim Radiosender „88 Radio Berlin“ erst bekannt, dass der Verkehr nur stockend zu fließend scheint. Wie der Sender von der Störung erfahren hat, ist nicht bekannt. Warum nur eine Geschwindigkeit von 40 km/h herausgeben wird ebenso nicht. Entsprechend zeigt sich ein Muster im umgekehrten Fall, wenn sich der Verkehr zu erholen scheint. Die FC-Daten zeigen im ersten Abschnitt bereits um 9 Uhr eine klare Verbesserung der gefahrenen Geschwindigkeit an, TMC erst einer Stunde später. Ausgehend von der daraus abgeleiteten Prämisse, dass FCD einen höheren Echtzeitanspruch hat als TMC, kann davon ausgegangen werden, dass FCD ein genaueres Bild der aktuellen Verkehrslage zeigt als TMC. Die TMC-Meldungen werden zu lange vom Sender ausgestrahlt, da erst eines der folgenden Kriterien auftreten muss, bevor die TMC-Meldung nicht mehr gesendet wird: Entweder die Gültigkeit der Meldung ist abgelaufen oder eine der TMC-Quellen, wie z.B. ein Hörer, meldet freie Fahrt. Da die Anzahl der Quellen bei FCD im gesamten Berliner Raum höher als bei TMC ist, ergibt sich dieser Unterschied auch bei anderen Strecken. Der zweite auffällige Punkt ist der Unterschied der Geschwindigkeiten. TMC zeigt eine Meldung mit 40 km/h an, dies ist im besten Fall z.B. 7:45 immer noch 15 km/h zu viel. Ausgehend von der zuvor beschriebenen Trägheit des Systems gegenüber FCD, kann die Meldung auch weniger genau sein, da die Realität zum genauen Zeitpunkt unbestimmt ist. Demnach sind zu keinem Zeitpunkt die aktuellen Geschwindigkeiten bekannt, sondern nur eine Tendenz bzw. eine übergeordnete Aussage, wie stockender oder zähfließender Verkehr. Dies kann auf der betrachteten Strecke einer Geschwindigkeit von 30 km/h entsprechen oder auch 50 km/h. Diese Information ist nicht bekannt. Somit muss für die Erstellung der TMC-Meldung ein sinnvoller Kompromiss gefunden werden, z.B. 40 km/h oder 45 km/h.

Diese Gegenüberstellung der TMC- und FC-Daten zeigt sehr gut die Schwächen des TMC-Systems, die durch FCD ausgeglichen werden können. Wie eine sinnvolle Fusion der Daten zu einer neuen und verbesserten Verkehrslage durch Nutzung der Stärken beider Systeme, führen kann, zeigt das folgende Kapitel 7.

7. Generierung einer neuen Datenbasis

Dieses Kapitel beschreibt die Erstellung einer neuen Übersicht der Verkehrslage von Berlin. Dabei soll eine sinnvolle Synergie zwischen den FC- und den TMC-Daten hergestellt werden. Wie und in welcher Form dies möglich ist, soll im Folgendem beschrieben werden.

7.1. Abgleich der Daten

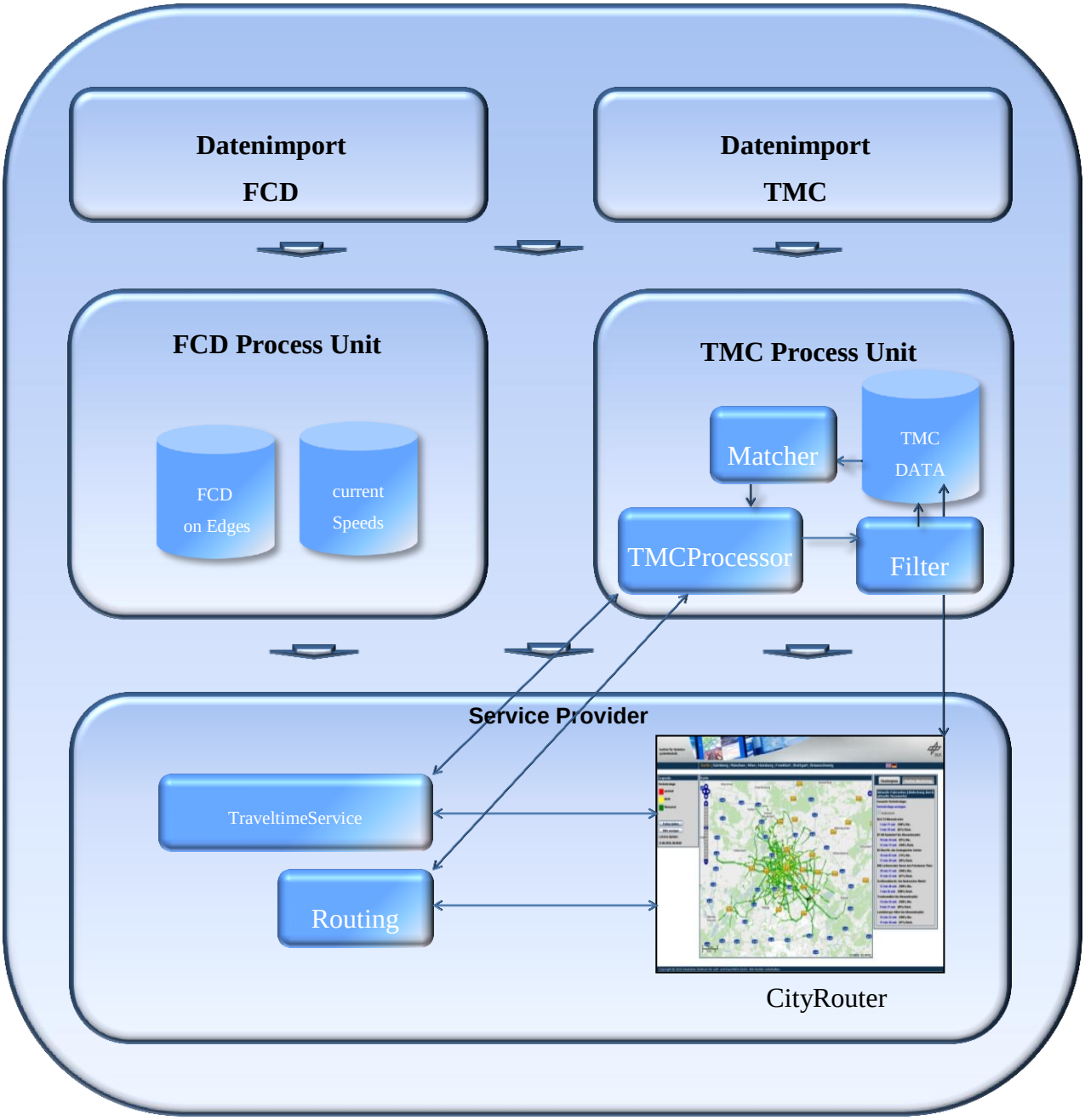


Abbildung 60: TMC & FCD Lösung

Die Abbildung 60 zeigt das aus Abbildung 31 bekannte Bild des Lösungskonzeptes für den Abgleich der TMC- und FC-Daten in seiner abgewandelten bzw. fertigen Lösungsform. Das zuvor erstellte Konzept deutete auf eine Kommunikation des „TMCProcessors“ mit einer der Tabellen hin, die von der „FCD Process Unit“ erstellt werden. Diese Schnittstelle entfällt und wird durch die Services „Traveltime“ und „Routing“ ersetzt. Wie der Prozessablauf des Abgleiches im „TMCProcessor“ intern verläuft, soll der Abschnitt 7.2. „ Prozessablauf“ detailliert zeigen. Zuvor soll eine Betrachtung der Möglichkeiten des Abgleiches zwischen TMC und FCD zu einer umfassenden Analyse führen.

Der Datenabgleich zwischen TMC und FCD ist grundsätzlich in beide Richtungen möglich. So könnten beide Systeme sich zwar ergänzen, aber nicht im gleichen Maße. Das ist in der Fehlerhäufigkeit von TMC bzw. FCD begründet. Die FC-Daten enthalten nur einen sehr geringen Fehler, im Sinne einer Falschmeldung. Die einzige Fehlerquelle im FCD-System sind die Bus- und Taxispuren, die an stark befahrenden Straßen den Bussen und Taxis eine Ausweichspur ermöglichen, die für die anderen Verkehrsteilnehmer verboten ist. Somit ist es möglich, dass ein Taxi unter Nutzung solch einer Fahrspur an einer Verkehrsstörung vorbei fährt, ohne diese zu erfassen. Die relevanten Kanten wurden aus diesem Grund aus dem System entfernt und jede eingehende FCD-Meldung zu diesen Strecken wird nicht weiterverarbeitet. Der Fehler der entstehen kann ist, dass eine von diesen nicht zu betrachtenden Strecken nicht aus dem System entfernt wurde, da keine Kenntnis über eine Ausweichspur bekannt ist bzw. diese neu ist oder nur vergessen wurde. DLR intern wird die Zuverlässigkeit der FC-Daten mit über 99.9% angegeben. [55] Für die TMC-Meldungen konnte bis jetzt keine ähnliche, wissenschaftlich fundierte Aussage erhoben werden. Somit ist bei einem Abgleich der TMC- mit den FC-Daten der FCD-Information immer Vorrang zu geben. FCD ist immer als richtig zu betrachten, wenn die Daten keine Verkehrsstörung anzeigen, TMC jedoch auf eine Störung hinweist. Der umgekehrte Fall beschreibt, dass die FC-Daten eine Verkehrsstörung anzeigen und TMC keine Meldung zu diesem Bereich vergeben hat. In diesem Szenario ist ebenfalls FCD zu vertrauen und eine etwaige Störung als richtig zu erachten. Aus diesem Grund soll eine potentielle TMC-Meldung gegen die FC-Daten validiert werden und nicht umgekehrt. Diese Forderung resultiert aus der unbestimmten Fehlerhäufigkeit der TMC-Meldungen und der Tatsache, dass die Daten an andere Unternehmen weiter gegeben und zur Forschung genutzt werden. Als Bedingung muss der Anteil der fehlerhaften Aussagen innerhalb der Verkehrslage so gering wie möglich gehalten werden, um auch weiterhin aussagekräftige Analysen und Prognosen erstellen zu können. Wie

der Abgleich der TMC- mit den FCD-Meldungen im Detail vollzogen wird, zeigt der folgende Abschnitt.

7.2. Prozessablauf

Im Folgendem soll beschrieben werden wie und welche TMC-Meldungen mit dem FCD-System abgeglichen werden können. Der rote Faden durch diesen Abschnitt soll der Prozessablaufplan aus Abbildung 61 sein. Dieser beschreibt den Umgang einer zu validierenden TMC-Meldung vom Eingang der Daten über dessen Möglichkeiten innerhalb des Prozesses bis hin zur persistenten Speicherung. Dies ist nur möglich, da die Validierung der Daten ein immer wiederkehrendes Szenario beschreibt, an dessen Anfang der Eingang einer neuen TMC-Meldung steht.

Direkt nach dem Eingang der TMC-Meldung wird diese auf ihr Ereignis hin überprüft. Wenn das Event einen Stau oder stockenden Verkehr beschreibt, kann es für einen FCD-Abgleich in Betracht gezogen werden. Wenn dem nicht so ist, handelt es sich um eine nicht geschwindigkeitsbezogene TMC-Meldung, wie z.B. eine Baustelle oder Fahrbahnverengung, welche sofort in der Datenbank gespeichert werden kann (siehe orange Raute, Abbildung 61, „Ist Meldung Geschwindigkeitsbezogen?“). Lautet die Antwortet „Ja“, handelt es sich um eine Meldungen, deren Event auf einen Stau oder stockenden Verkehr hinweist. In diesem Fall wird die Meldung dem „TMCValidatorFCD“ des „TMCProcessors“ übergeben. Diese eigenständige Einheit überprüft, ob FC-Daten im System für die Position der TMC-Meldungen vorliegen. Hierzu wurde das Navteq[45] Straßennetz beim Start der „TMC Process Unit“ eingelesen und als Objekte der Klasse Links (Kanten) und Nodes (Knoten) in den HashMaps „hashMapStreetNetworkLinks“ und „hashMapStreetNetworkNodes“ gespeichert. Dabei referenziert jeder Knoten eine Position in Länge und Breite und jede Kante einen Start- und Endknoten. Die zu bearbeitende TMC-Meldung besitzt ebenfalls eine Längen- und Breiteninformation, die aus dem Location-Code gewonnenen Informationen. Ein „Matcher“ sucht anschließend auf Basis des Navteqnetzes den nahegelegensten Knoten zur TMC-Meldung. Hierzu iteriert dieser über alle Knoten aus dem NodesSet der HashMap „hashMapStreetNetworkNodes“. Die dazugehörige Berechnung ist eine einfache Abstandsbestimmung, bei der im Betrag die kleinste Differenz zwischen Breitengrad der TMC-Meldung und Breitengrad des Navteqnetzes sowie die kleinste Differenz des jeweiligen

[45] Navteq ist ein Anbieter von Geodaten und des darauf aufbauenden Kartenmaterials. [9]

Längengrades bestimmt wird. Dieses Verfahren wird ebenfalls auf den ersten Extent der TMC-Meldung angewendet, so dass anschließend 2 zugehörige Punkte im Navteqnetz bekannt sind, die das TMC-Ereignis in seiner Ausdehnung beschreiben. Diese beiden Punkte werden dem „Routing“ Service übergeben, der als Antwort eine Liste mit Kanten-ID’s zurückgibt. Diese ID‘s werden in der Tabelle „tdp_fcd_on_edges“ abgefragt und anschließend wird aus der Spalte „local_time“ ersichtlich, wann das letzte Fahrzeug Daten für diese Strecke erhoben hat. Wenn die Daten nicht älter als eine halbe Stunde sind, ist FCD mit aktuellen Daten verwendbar und die TMC-Meldung kann abgeglichen werden. Siehe Abbildung 61 in der Verzweigung „Ist FCD aktuell?“.

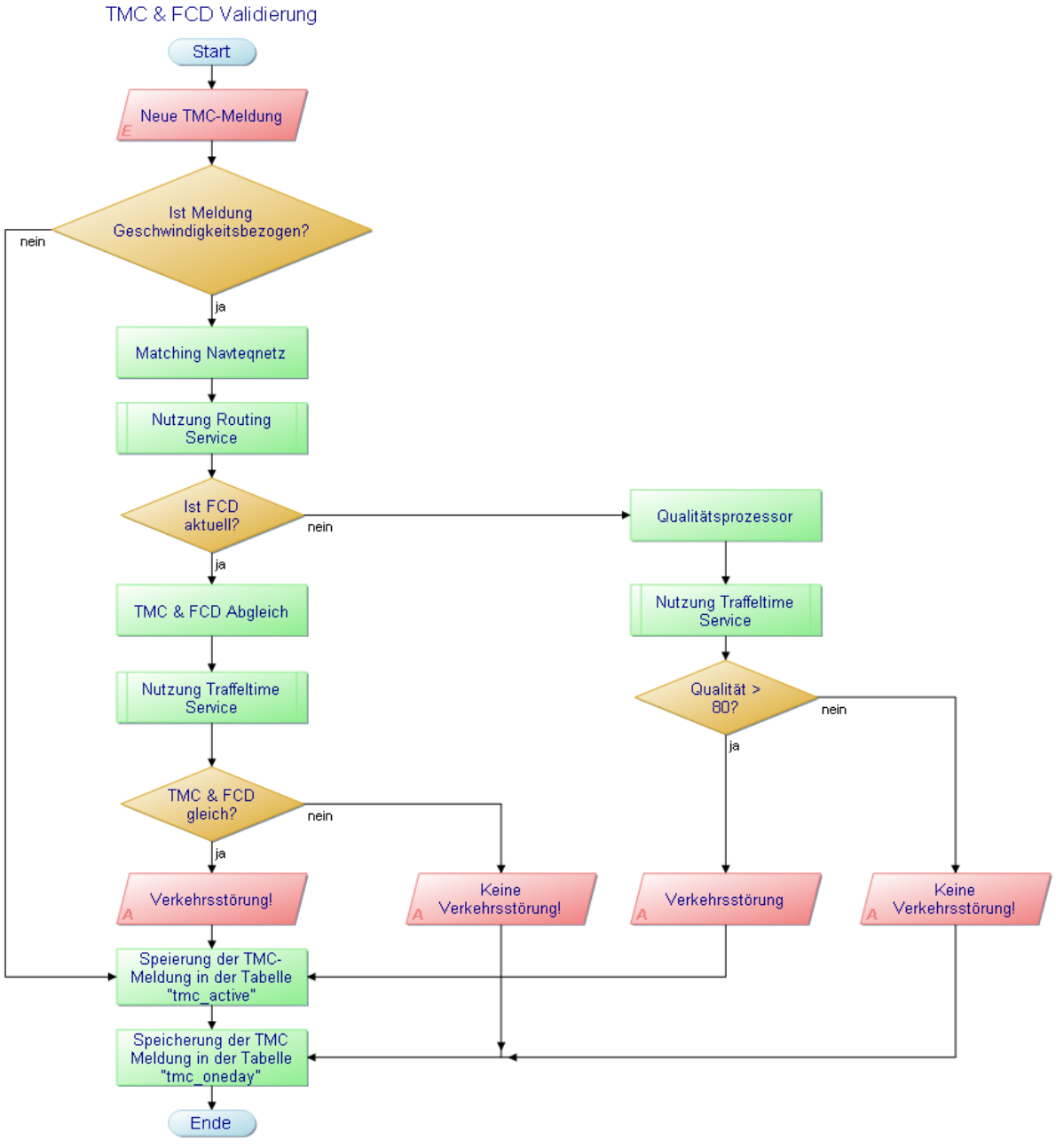


Abbildung 61: Prozessablaufplan TMC & FCD Validierung

Folgend soll der linke Prozessablauf näher betrachtet werden. Die TMC-Meldung kann mit aktuellen FC-Daten abgeglichen werden, da der zuvor beschriebene „Matcher“ verwertbare Daten im System gefunden hat. Der eigentliche Abgleich verwendet den Traffeltime Service, der wiederum die Kanten-ID’s als Eingabe erwartet. Die Rückgabe ist ein Objekt vom Typ „TraffelTimeInfo“, welcher den Zugriff auf die erlaubte und aktuelle Geschwindigkeit ermöglicht. Wenn die aktuelle Geschwindigkeit die erlaubte um die Hälfte unterschreitet, liegt auch im FCD-System eine Verkehrsstörung vor und die Verzweigung „TMC & FCD gleich?“ würde mit „Ja“ zu beantworten sein. Gemeint ist, dass FCD und TMC die gleiche Aussage treffen bzw. das FCD die TMC-Meldung bestätigt. In diesem Fall kann die Meldung in der Datenbank abgelegt werden. Beschreibt FCD keine Störung, wird die TMC-Meldung als Fehlmeldung interpretiert und nicht in der Tabelle „tmc_active“ sondern nur in der Tabelle „tmc_oneday“ gespeichert. Dieser Ablaufplan beschreibt das Vorgehen, wenn aktuelle FC-Daten verwendet werden können. In Folgenden soll jetzt beschrieben werden, wenn keine aktuellen FC-Daten im System für die Position der TMC-Meldung vorliegen.

Wenn es keine Möglichkeit gibt, auf vorhandene aktuelle FC-Daten zurückzugreifen, wird die Meldung an den Qualitätsprozessor übergeben. Dieser bestimmt zuerst die aktuelle Tageszeit mit Hilfe der Java Klasse „Date“. Diese Zeit wird auf die jeweilige Ganglinie der Position übertragen.

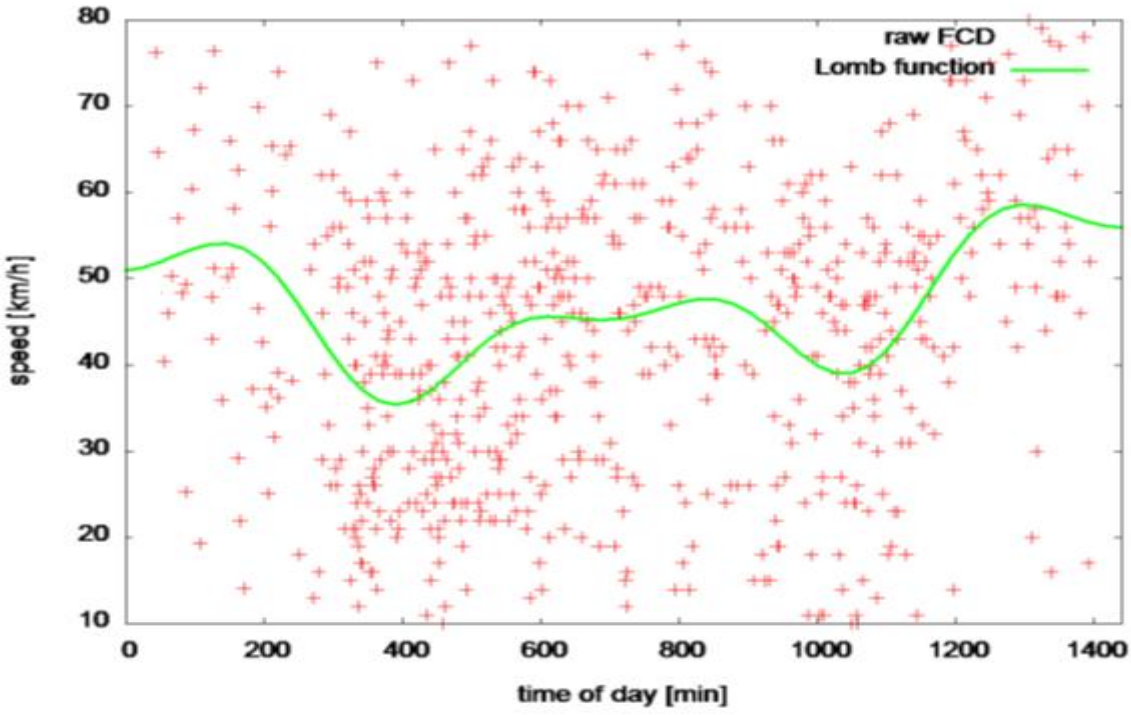


Abbildung 62: Ganglinie

Die Abbildung 62 zeigt eine vom DLR erstellte Ganglinie vom 10.11.2010 für die Kante mit der ID 51931707 (Schwarnweberstraße, Ecke Antonienstraße bis Schwarnweberstraße, Ecke Eichborndamm). In der Abbildung wird die gefahrene Geschwindigkeit in Kilometer pro Stunde (X-Achse) über der Tageszeit in Minuten (Y-Achse) dargestellt. Auf dieser Strecke ist nach StVO 50 km/h erlaubt. Die roten Kreuze zeigen alle aufgenommenen Daten für diesen Abschnitt. Die grüne Linie zeigt die erstellten Werte einer DLR internen Funktion, die verallgemeinert, die mittlere Verteilung der Werte für den jeweiligen Tagesabschnitt repräsentiert, die sogenannte Ganglinie. Das genaue Erstellungsverfahren gehört zum Know-how des Instituts und soll nicht weiter beschrieben werden. [55] Als Interpretation für die Ganglinie wird das zu erwartende normale Geschwindigkeitsverhalten für diesen Abschnitt an einem typischen Tag, in diesem Fall ein Mittwoch, angenommen.

Auffällig ist, dass ab Minute 180 bzw. ab 3 Uhr morgens, die Geschwindigkeit in diesem Bereich bis 6:40 Uhr, oder Minute 400, stetig abfällt und auch im weiteren Tagesverlauf bis 20 Uhr unter der erlaubten Geschwindigkeit von 50 km/h bleibt. Dies ist eine eher schwach befahrene Straße, da die Ganglinie nicht unter die Hälfte der erlaubten Geschwindigkeit fällt bzw. somit auf keine Verkehrsstörung hinweist.

Diese Kurven (Ganglinien) werden auf Basis der historischen Daten im System erstellt und ständig ergänzt. Beim Eingang einer TMC-Meldung, bei der auf die historischen Daten zurückgegriffen werden muss, wird durch die jeweilige Ganglinie eine horizontale Linie gezogen, die der Hälfte der erlaubten Geschwindigkeit entspricht. Unterschreitet die Ganglinie diese Linie zum Zeitpunkt des Einganges einer in Frage kommenden Meldung, wird diese mit der Qualität größer 80 gekennzeichnet. Die genaue Qualitätsbestimmung zeigt als Beispiel das folgende Diagramm.

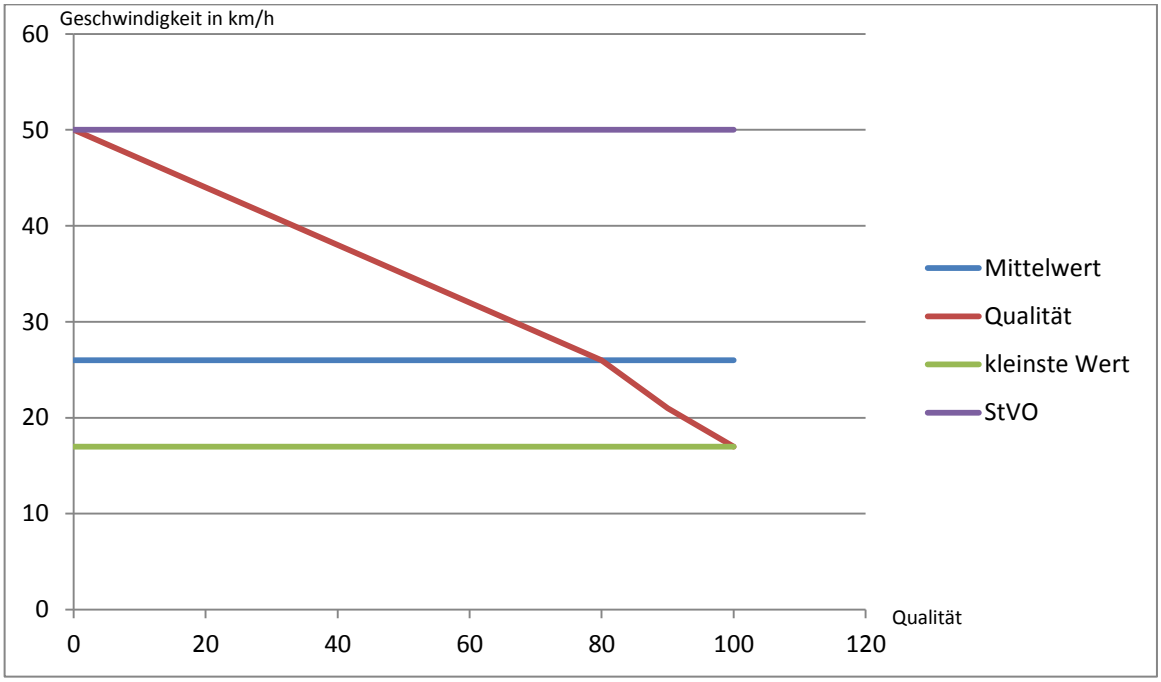


Diagramm 5: Qualitätsbestimmung

Das Diagramm 5 zeigt, wie die Qualitätsbestimmung funktioniert. Dabei dient das Diagramm lediglich der Erklärung der ablaufenden Funktionen und wird nicht zur Laufzeit in dieser Form erstellt.

Jeder Streckenabschnitt hat eine nach StVO vorgeschrieben Geschwindigkeit, z.B. 50 km/h, (lila Linie) und einen Stau Wert (grüne Linie) sowie einen Mittelwert (blaue Linie), der der Hälfte der erlaubten Geschwindigkeit entspricht, also in diesem Beispiel 25 km/h. Die Bestimmung der blauen Linie leitet sich direkt aus den Vorgaben des DLR für das FCD ab, siehe Tabelle 5 „Färbungstabelle“. Zur Erstellung der grünen Linie wird der Mittelwert aller Werte genommen, die zwischen 0 und der Hälfte der erlaubten Geschwindigkeit liegen, der blauen Linie. Dieser Wert wird auch als F-Wert bezeichnet und leitet sich aus dem „Level of Service“ ab. Der Wert beschreibt eine Verkehrsdichte, die die Kapazitäten der Straße und der steuernden Elemente zu den dort befindlichen Fahrzeugen übersteigt. [8][S154]

Des Weiteren zeigt die rote Linie im Diagramm die Bestimmung der Qualität bzw. die Funktion der zu bestimmenden Geraden. Dabei ist zu sehen, dass die rote Linie eigentlich aus zwei Geraden besteht, die die alle Werte oberhalb der blauen Linie und alle Werte unterhalb der blauen Linie beschreibt. Beim Ablesen der Ganglinie zum Zeitpunkt der eintreffenden TMC-Meldung bestimmt die Qualitätsbestimmung zuerst, ob die Ganglinie sich über oder unter der blauen Linie befindet. Anschließend wird je nach Entscheidung die entsprechende Funktion genutzt.

Die Erstellung der beiden Funktionen richtet sich ebenfalls nach den Vorgaben des DLR. Diese beschreiben, dass jede Meldung die zum Zeitpunkt ihres Eintreffens die blaue Linie unterschreitet, eine Qualität größer 80 haben soll, da das DLR intern für andere Verarbeitungsprozesse im FCD-System nur Werte mit einem Qualitätsmerkmal größer 80 weiter verarbeitet. Um eine möglichst große Kompatibilität für spätere Verarbeitungsprozesse zu sichern, soll auch bei der TMC-Verarbeitung ein Qualitätswert größer 80 eine Weiterverarbeitung bedingen. Demnach hat der historische Geschwindigkeitswert der jeweiligen Ganglinie zu einem Zeitpunkt, bei dem die Hälfte der erlaubten Geschwindigkeit unterschritten wird, immer den Qualitätswert von mindestens 81. Im obigen Beispiel entspricht dies dem Schnittpunkt (P_S = (80, 25)) der roten und der blauen Linie. Die obere Gerade enthält somit die beiden Punkte P_1 = (0, 50) und P_2 = (80, 25), abzulesen aus dem Diagramm. Zur Bestimmung der Geradengleichung soll zuerst die Steigung der Geraden bestimmt werden. Allgemein beschrieben mit: [8]S85

$$m = \frac{\Delta y}{\Delta x} = \frac{\mathrm{y2} - \mathrm{y1}}{x2 - x1}$$

Für die Punkte P_1 = (0, 50) und P_2 = (80, 25) ergibt sich so:

$$m = \frac{-25}{80} = \frac{25 - 50}{80 - 0}$$

Der Schnittpunkt mit der Y-Achse wird berechnet nach: [8]S85

$$n = \mathrm{y1} - \frac{y2 - y1}{\mathrm{x2} - \mathrm{x1}}\, x1$$

Für die Punkte P_1 = (0, 50) und P_2 = (80, 25):

$$n = 50 - \frac{25 - 50}{80 - 0}\, 0 = 50$$

Die Werte für n und m werden in die allgemeine Geradengleichung eingesetzt: [8]S85

$$y = \mathrm{mx} + \mathrm{n}$$

und beschreiben die Gerade in der Form:

$$y = \frac{-25}{80}\mathrm{x} + 50$$

Für die zweite Gerade wird anlog verfahren, wobei die Punkte der Geraden sich verändern

P_1 = (80, 25) und P_2 = (100, 17). Die zugehörigen Gleichungen sind:

$$m = \frac{-8}{20} = \frac{17 - 25}{100 - 80}$$

$$n = 25 - \frac{17 - 25}{100 - 80}\ 80 = 57$$

$$y = \frac{-8}{20}\mathrm{x} + 57$$

Allgemein werden die Punkte der Geradengleichung über der blauen Linie immer wie folgt bestimmt: P_1 = (X_1, Y_1), wobei X_1 immer den Wert 0 hat, da dies der Startwert der Qualitätsskala ist. Der Y Wert wird immer durch dem Geschwindigkeitswert nach StVO für diesen Abschnitt festgelegt, wie aus dem Diagramm ablesbar ist (lila Linie). Der zweite Punkt P_2 = (X_2, Y_2) wird immer beschrieben mit X_2 = 80 und Y_2 = die Hälfte der erlaubten Geschwindigkeit, wie es vom DLR vorgegeben wird. Für die Funktion unter der blauen Linie gilt: P_1 = P_2 aus der Geraden oberhalb der blauen Linie und P_2 = (X_2, Y_2) mit X_2 = 100, dem maximalen Wert der Qualitätsskala und Y_2 = der F-Wert. Die Punkte werden in diese Funktion eingesetzt:

$$f(x) = (\frac{\mathrm{y2-y1}}{x2-x1})\ \mathrm{x} + (\ \mathrm{y1} - (\frac{y2-y1}{\mathrm{x2-x1}})\ x1)$$

Umgestellt für den Qualitätswert X:

$$x = (\mathrm{y} - \mathrm{y1})\ \frac{(x2 - x1)}{(\mathrm{y2} - \mathrm{y1})} + x1$$

Die erhaltene Gleichung beschreibt den Qualitätswert X der TMC-Meldung. Somit liefern die Funktionen den Qualitätswert X für eine Geschwindigkeit Y. Wichtig ist, dass die Funktionen für jede TMC-Meldung, die geschwindigkeitsbezogen ist und nicht mit aktuellen FC-Daten abgeglichen werden kann, nur einmal pro Tag erstellt werden muss.

Der Qualitätsprozessor beschreibt nun auf Basis der inneren Funktionen die TMC-Meldung. Bei einem Wert kleiner-gleich 80 wird die TMC-Meldung als Fehlmeldung interpretiert und nicht in der Tabelle „tmc_active“ sondern nur in der Tabelle „tmc_oneday“ gespeichert. Bei einem Wert größer 80 wird die Meldung auch in der Tabelle „tmc_active“ für die Visualisierung gespeichert, da die historischen FC-Daten eine Verkehrsstörung nahelegen.

7.3. Neue Übersicht der Verkehrslage

Die neue Datenbasis von Berlin besteht aus den FC- und den TMC-Daten. Dabei werden die TMC-Meldungen mit den FC-Daten, wie gezeigt, abgeglichen. Doch welche Möglichkeiten sich aus den neuen Daten ergeben, soll in diesem Abschnitt gezeigt werden. Des Weiteren soll eine Unterscheidung in Quantität und Qualität bei der Betrachtung eine Rolle spielen.

Quantitative Beurteilung

Bei genauerer Betrachtung der in der Arbeit bestimmten Zahlen drängt sich die Frage auf, wie relevant sind die TMC-Meldungen für die Verkehrslage. Aus der Arbeit ist bekannt, siehe 3.3.2. „Datenerhebung“, dass das FCD-System 30 Millionen Daten im Monat für den urbanen Bereich Berlin generiert und nach der TMC-Datenanalyse das TMC an einem normalen Montag beim Sender „Radio Berlin“ in etwa 346 neuen Meldungen pro Tag ausgibt. Auch wenn die TMC-Meldungen auf Grund ihrer unbestimmten Fehlerrate eine unbestimmte Menge an tatsächlichen neuen TMC-Meldungen für jeden Tag generieren, ist bei einer sehr großzügigen Betrachtung von 500 neuen Meldungen am Tag der Monatswert selbst bei 31 Tagen nur bei 15500 neuen Meldungen. Das würde weniger als 1 Prozent von 30 Millionen FC-Daten entsprechen. Selbst bei noch großzügigerer quantitativer Betrachtung der TMC-Meldungen, würde der Anteil der TMC-Meldung weit unter 1 Prozent liegen. In diesem Vergleich zeigt sich, dass TMC quantitativ nur sehr wenig zu der Datenbasis beiträgt.

Qualitative Beurteilung

Bei der Qualitätsbeurteilung soll gezeigt werden, wie wichtig die TMC-Meldungen für die Verkehrslage sind. Wie mehrfach beschrieben, gibt FCD nur Auskunft über eine Geschwindigkeit an einen bestimmten Punkt. Dies reichte bisher um eine geschwindigkeitsbezogene Verkehrslage zu erstellen. Jedoch konnte bis jetzt nicht bestimmt werden, warum hier eine Verkehrsstörung vorliegt. Die TMC-Daten ergänzen das bestehende System genau um diese Informationen, somit ist es erstmals möglich eine Verkehrsstörung mit einem Ereignis in Zusammenhang zu bringen.

Abbildung 63: Ereignis für FCD

Die Abbildung 63 zeigt die Behrenstraße, Ecke Friedrichstraße in Berlin. Wie am FCD-Layer zu sehen (gelbe Linien), liegt eine Verkehrsstörung vor. Vor der Erstellung der neuen Datenbasis konnte der Nutzer an dieser Stelle erkennen, dass die Geschwindigkeit beeinträchtigt ist. In Folge der neuen Daten kann erstmals die Störung mit dem Ereignis einer Baustelle in Zusammenhang gebracht werden. Wie diese Anzeige sich zusammenstellt und welche Möglichkeiten sich für den Endnutzer ergeben, soll im Kapitel 8. „Darstellung der neuen Datenbasis“ genau beschrieben werden.

Die TMC-Daten ergänzen die Verkehrslage um eine völlig neue Größe, die der Ereignisse. Die daraus resultierenden Möglichkeiten der Verkehrsanalyse, soll im Kapitel 11. „Resultate“ noch einmal perspektivisch aufgegriffen werden.

8. Darstellung der neuen Datenbasis

Aus den vorrangegangen Kapiteln wird ersichtlich, wie TMC und FC-Daten empfangen, verarbeitet und gespeichert werden. Ferner wurde beschrieben, wie TMC auf Basis von FCD überprüft werden kann und was die neue Übersicht der Verkehrslage kennzeichnet bzw. welchen Neuheitsgrad sie besitzt. In diesem Kapitel soll beschrieben werden, wie diese neue Übersicht auf dem erstellten Webfrontet des Cityrouter dargestellt werden kann und welche Architekturen und Mechanismen eine sinnvolle Kopplung zwischen Anzeige und Daten sicherstellen.

8.1. GWT (Grundlagen für TMC Service)

Bevor die Daten angezeigt werden können, müssen die Daten aus der Datenbank geholt werden. Wie im Kapitel 5. Abschnitt 2.4. beschrieben wurde, enthält die Tabelle „tmc_active“ die aktuellen TMC-Meldungen bzw. jene Meldung, die angezeigt werden soll. Bevor das Verfahren detailliert beschrieben werden kann, soll kurz auf die für die Darstellung von TMC wichtigen Grundlagen von Google Web Toolkit (GWT) eingegangen werden, da mit dessen Hilfe das Web Frontend des Cityrouters programmiert wurde.

Das Google Web Toolkit, kurz GWT, ist ein Werkzeug zur Entwicklung von Webanwendungen. Der Kern besteht aus einem Compiler, der es einem Java Entwickler ermöglicht, Java-Code in Javascript zu übersetzen, was wiederum von einem Browser angezeigt werden kann. Des Weiteren gibt es zahlreiche Komponenten zur Darstellung von Hintergründen, Buttons und Menüs usw. Ferner ist es einfach an das modulares System anzuknüpfen, um eigene Bibliotheken zu erstellen, wie z.B. die der Gemeinschaft von OpenStreetMap. Dieses freie Kartenmaterial ermöglicht das Erstellen von eigenen Karten, wie es beim Cityrouter eigens verwendet wird.

Abbildung 64: Karte Cityrouter

Die Abbildung 64 zeigt die erstellte Karte als einen Ausschnitt aus der Kartenanzeige des Cityrouter. Die Straßen sind betont monochrom in Grautönen gehalten und sollen so von der angezeigten Verkehrslage nicht ablenken. GWT unterstützt die Anzeige und das Handling der gezeigten Karte und bietet somit dem Nutzer die Möglichkeit, zu interagieren mittels Zoomen, Ziehen oder Bewegen. Darüber hinaus beschreibt GWT Möglichkeiten der Server Client Kommunikation über Remote Procedure Call (RPC). Dieser Fernaufruf einer Methode vom Clienten zum Server ermöglicht die Interaktion auf der Ebene von serialisierbaren Java-Objekten. Serialisierbar werden Objekte genannt, bei denen für eine Menge von Transaktionen, z.B. die zu übertragenen Bytes, eine Festlegung für eine Reihenfolge sämtlicher relevanter Operationen möglich ist. Auf die Erstellung solch eines Aufrufes wird im Folgenden noch genauer eingegangen. Weitere wichtige Möglichkeiten von GWT sind die enthaltene Debug-Umgebung, die Unterstützung der Browser Safari, Chrome, Mozilla Firefox und Internetexplorer, die integrierte Testumgebung JUnit und das Einbinden von Nativen Codes, wie C, C++, JavaScript oder einfaches HTML.

Für das weitere Verständnis der Serviceimplementierung soll der Screenshot aus Abbildung 65 ein wesentliches Merkmal jedes GWT Projektes verdeutlichen. Die Pakete tragen alle den ihnen zugewiesenen Namen mit der Kennung „de“ für Deutsch, „dlr“ für die Firma, „cityrouter“ für den Projektname. Die folgenden Bezeichner unterscheiden sich von Paket zu Paket. Dabei deutet „client“ darauf hin, dass dieses Paket beim Clienten ausgeführt wird und „server“, dass es serverseitig ausgeführt wird. Beim Deployment[46] auf dem Server achtet GWT selbst auf die Bereitstellung bzw. die Ausführung des Codes auf der jeweils richtigen Seite der beiden Kommunikationspartner.

[46] Deployment beschreibt in der Informatik den Ablauf zur Installation einer Software.

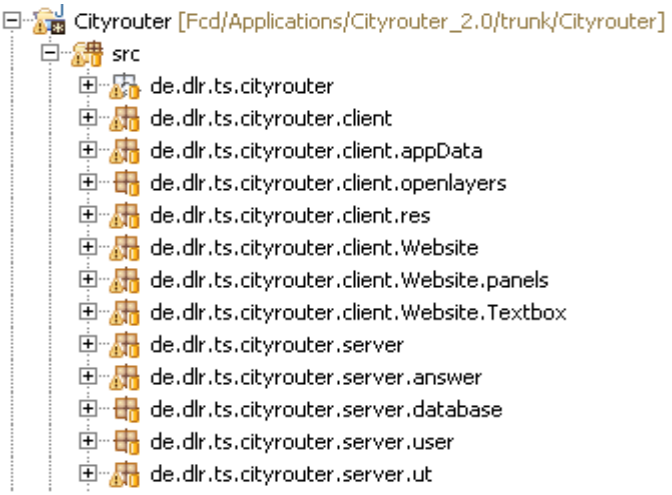


Abbildung 65: Paketansicht Cityrouter

8.2. TMC serverseitig

Wie in dem kurzen Überblick zu GWT gezeigt wurde, gibt es bei Anwendung dieser Art einen Teil, der vom Server ausgeführt wird und einen, der beim Nutzer, dem Client, ausgeführt wird. Bevor der Client eine Anfrage an den Server stellen kann, um sich die aktuellen TMC-Meldungen geben zu lassen, muss der Server die Daten selbst besitzen. Hierzu wurde ein eigener Datenbank-Händler entworfen, der mit der bereits bekannten Datenbankschnittstelle JDBC[47] arbeitet. Diese Verbindung erstellt Strings von den Spalten „lat“, „lon“, „frequency“ und „wDecode“ aus der Tabelle „tmc_active“. Deren Inhalt wurde bereits im Kapitel 5. Abschnitt 2.4. „Speicherung“ beschrieben. Die Erstellung eines TMC-Objektes auf der Serverseite der Anwendung nach dem Vorbild des TMC-Objektes der „TMC Process Unit“ ist aus Performancesicht nicht zu empfehlen. Auch wenn die Erstellung eines solchen Objektes sich aus Verwaltungs- und Bearbeitungssicht anbietet, darf nicht vergessen werden, dass die Übertragung mehrerer Objekte aufwendiger ist und länger dauert, als die einer einzelnen, langen Zeichenkette. Die Informationen aus den benannten Spalten reichen, um die geforderten Informationen vom DLR dem Nutzer zu Verfügung zu stellen. Dies ist die Position des Ereignisses (lat, lon), die Quelle des Datensatzes und Inhalt, das eigentliche Ereignis und dessen Decodierung. Doch bevor die Methode „getTMCString“ der Klasse „TMCDBHandler“ eine Zeichenkette dem Client auf eine Anfrage hin übergeben kann, muss diese zum Parsen aufgearbeitet werden. Dazu werden die Spalteninformationen in einer TMC-Meldung mit einem Semikolon „ ; “ getrennt. Die Meldungen selbst werden durch Doppelkreuz „ # “ voneinander getrennt.

[47] Java Database Connectivity (JDBC) ist eine einheitliche Schnittstelle für die Kommunikation mit verschiedenen Datenbanken in Java.

```
     * @return tmcString
     */
    public String getTmcString() {

        String tmcstring = "";
        try {
            if (conn == null || conn.isClosed()) {

                conn = DriverManager
                        .getConnection(getUrl(), dbUser, dbPassword);
            }
            Statement query = conn.createStatement();

            String sql = getSqlInsertString();
            ResultSet resultSet = query.executeQuery(sql);

            while (resultSet.next()) {

                String lat = resultSet.getString("lat");
                String lon = resultSet.getString("lon");
                String rds = resultSet.getString("rds");
                String frequency = resultSet.getString("frequency");
                String weventsDecoded = resultSet.getString("wDecode");

                tmcstring += lat + ";" + lon + ";" + rds + ";" + frequency
                        + ";" + weventsDecoded + "#";
            }
            query.close();

        } catch (SQLException e) {
            e.printStackTrace();

        }
        return tmcstring;
    }
```

Abbildung 66: TMCDBHandler

Die Abbildung 66 zeigt die Methode „getTMCString“ und deren Funktionalität. Nach einer Überprüfung des JDBC-Verbindungsobjektes „conn“ wird eine Abfrage über die benannten Spalten getätigt. Das Ergebnis dieser Abfrage wird in der beschriebenen Form als eine zusammenhängende Zeichenkette erstellt.

Damit der Server auf eine Anfrage des Client reagieren kann, benötigt dieser einen Service, den sogenannten „RemoteService“. Dazu wird ein neuer Service für den Cityrouter erstellt. Die Klasse „CityrouterService“ repräsentiert die Schnittstelle zwischen Client und Server, das sogenannte Interface erbt von der Klasse „RemoteService“ der GWT-API.

```
@RemoteServiceRelativePath("service")
public interface CityrouterService extends RemoteService {

    String tmcService(String input) throws IllegalArgumentException;
```

Abbildung 67: TMC Service

Die Abbildung 67 zeigt, wie beim Interface die Methode „tmcService“ der Signatur der Schnittstelle hinzugefügt wird. Da die Kommunikation auch asynchron bereitgestellt werden soll, muss dieser Service auch für diesen Teil der Schnittstelle hinzugefügt werden. Bei einer asynchronen Anfrage des Client wartet dieser nicht auf eine Antwort vom Server und der Nutzer kann die Anwendung weiter nutzen.

```
public interface CityrouterServiceAsync {

    void tmcService(String input, AsyncCallback<String> callback)
            throws IllegalArgumentException;
```

Abbildung 68: TMC Service asynchron

Die Abbildung 68 zeigt die Deklaration der Methode „tmcService“ für den asynchronen Teil. Der zweite Übergabewert ist ein Objekt der Klasse „AsyncCallback“, dieses verwaltet die eigentliche Anfrage. Die dazu verwendeten Mechanismen führen den vom Compiler erstellten JavaScriptCode auf eine AJAX[48] Architektur zurück. Das entspricht einem einfachen XMLHttpRequest. Das XMLHttpRequest ist wiederum eine API zum Transfer von Daten über das HTT-Protokoll.

Für die eigentliche Registrierung des Services wird in der Klasse „CityrouterServiceImpl“ die Methode „tmcService“ registriert.

```
@SuppressWarnings("serial")
public class CityrouterServiceImpl extends RemoteServiceServlet implements
        CityrouterService {

    @Override
    public String tmcService(String input) throws IllegalArgumentException {
        return TMCDBHandler.getInstance().getTmcString();
    }
```

Abbildung 69: TMC Service Impl

[48] Asynchronous JavaScripting and XML (AJAX) beschreibt die Technologie für eine asynchrone Datenübertragung zur Darstellung von dynamischen Webinhalten.

Die Abbildung 69 zeigt, dass die Klasse „CityrouterServiceImpl“ von der Klasse „RemoteServiceServlet“ erbt und die Schnittstelle CityrouterService implementiert. Für die Erfüllung der Signatur beschreibt die Klasse die Methode „tmcService“, in dem sie ihr einen Rückgabewert gibt. Dieser greift wiederum auf die Methode „getTMCString“ der zuvor beschrieben Klasse „TMCDBHandler“ zurück. Diese generiert den TMC-String, der wiederum vom Servlet als String an den Client übergeben werden kann.

8.3. TMC clientseitig

Bei jedem Start der Cityrouteranwendung durch einen Nutzer wird automatisch die Methode „tmcCreateService“ der Klasse „CityrouterClient“ aufgerufen.

```
/**
 * @param input
 */
public void tmcCreateService(final String input) {

        cityrouterService.tmcService(input, new AsyncCallback<String>()

        @Override
        public void onFailure(Throwable caught) {

        }

        @Override
        public void onSuccess(String tmcString) {
            openlayers.createTMCLayer(tmcString);
        }

    });
}
```

Abbildung 70: TMC Client Anfrage

Die Abbildung 70 zeigt die Methode „tmcCreateService“, die bei jeden Aufruf und alle 5 Minuten die Methode „tmcService“ der Schnittstelle „CityrouterService“ aufruft, welche selbst die asynchrone Schnittstelle verwendet. Dies führt zu einem Durchreichen der Anfrage zum Servlet der Klasse „CityrouterServiceImpl“, das wie bekannt ist, direkt mit dem Datenbank-Händler des Servers kommuniziert.

Bei einem fehlerhaften Aufruf wird beim Client die Methode „onFailure“ aufgerufen, die, wie in der Abbildung zu sehen ist, keine weiteren Aktionen ausführt. Dies ist nicht notwendig, da der Nutzer über die misslungene Anfrage nicht benachrichtigt werden soll. Bei einer erfolgreichen Anfrage wird die Methode „onSuccess“ aufgerufen, die die Methode „createTMCLayer“ aufruft. Als Übergabe erwartet diese Methode den zuvor erstellten TMC-

String bzw. die Antwort des Servers. Die Erstellung des neuen Layers benötigt 2 Schritte, die in der Methode vollzogen werden müssen.

Der erste Schritt ist die Decodierung des übergebenen Strings.

```
public void createTMCLayer(String result) {

    String tmcMassage[] = result.split("#");

    for (int i = 0; i < tmcMassage.length; i++) {
        String tmcSplitData[] = tmcMassage[i].split(";");

        if (tmcSplitData.length > 3) {

            String key = trimTMCToLength5(tmcSplitData[1])
                    + trimTMCToLength5(tmcSplitData[0]);
            tmcHashMap.put(key, createTMCData(tmcSplitData[0],
                    tmcSplitData[1], tmcSplitData[2], tmcSplitData[3],
                    tmcSplitData[4]));
        }
    }
```

Abbildung 71: TMC-Layer 1

Die Abbildung 71 zeigt den ersten Teil der Methode „createTMCLayer“, die Decodierung des Antwort-Strings. Wie aus dem Abschnitt 8.2. „TMC serverseitig“ bekannt ist, werden die TMC-Meldungen mit Doppelkreuz getrennt und die Informationen innerhalb einer Meldung mit Semikolon. Wie die Abbildung 71 zeigt, werden die TMC-Meldungen getrennt in dem Feld „tmcMessage“ abgelegt. Anschließend wird jedes Feld ausgelesen und selbst wieder in das Feld „tmcSplitData“ aufgeteilt, so dass dieses Feld die einzelnen Elemente einer TMC-Meldung enthält. Diese einzelnen Elemente werden der Methode „createTMCData“ übergeben, die ein neues Objekt vom Typ „TMCData“ erstellt. Dieses Objekt entspricht der bekannten Interpretation aus Kapitel 5. Jede TMC-Meldung bzw. das erstellte Objekt wird der HashMap „tmcHashMap“ als Referenz seiner Position hinzugefügt. Die Fülle der TMC-Meldungen liegt zu diesem Zeitpunkt beim Client als Objekte von „TMCData“ vor und kann jetzt angezeigt werden.

Der zweite Schritt in der Methode „createTMCLayer“ beschreibt die Anzeige der TMCData Objekte.

```
        if (map.getLayerByName("TMC") != null)
            map.removeLayer(map.getLayerByName("TMC"));
        setTmcMarkers(new Markers("TMC"));

        setTmcEventsHandling(new TMCEventsHandling(this, map, getTmcMarkers(),
                tmcHashMap));
        getTmcMarkers().getEvents().register("mouseover", getTmcMarkers(),
                getTmcEventsHandling());
        getTmcMarkers().getEvents().register("mouseout", getTmcMarkers(),
                getTmcEventsHandling());
        getTmcMarkers().getEvents().register("mousemove", getTmcMarkers(),
                getTmcEventsHandling());

        // Iterate over the Map and create Marker to Layer
        for (Iterator<TMCData> iterator = tmcHashMap.values().iterator(); iterator
                .hasNext();) {
            TMCData tmcData = iterator.next();

            if (tmcData.getLon() == null || tmcData.getLat() == null)
                continue;

            LonLat lonlat = new LonLat(Double.parseDouble(tmcData.getLon()),
                    Double.parseDouble(tmcData.getLat()));
            lonlat.transform("EPSG:4326", "EPSG:900913");

            getTmcMarkers().addMarker(new Marker(lonlat, getTmcIcon(tmcData)));
            getTmcMarkers().redraw(true);
        }
        map.addLayer(getTmcMarkers());

    }
```

Abbildung 72: TMC-Layer 2

Die Abbildung 72 zeigt den 2-ten Teil der Methode „createTMCLayer“. Bevor die Daten angezeigt werden, wird überprüft, ob ein Layer Objekt mit dem Namen „TMC“ bereits existiert. Wenn dem so ist, soll der alte Layer gelöscht werden. Anschließend erstellt die Methode „setTMCMarkers“ einen neuen TMC-Layer mit dem Namen „TMC“. Das erstellte Objekt ist mit der Methode „getTMCMarkers“ referenzierbar. Bei dem noch leeren Layer werden die Ereignisse „mouseover“, „mouseout“ und „mousemove“ registriert. Was diese Ereignisse bewirken sollen, wird im Folgendem noch beschrieben. Nach dieser Registrierung wird über die HashMap „tmcHashMap“ iteriert. Beim Lesen aller „TMCData“ Objekte der HashMap wird von den Positionsdaten ein Objekt „LonLat“ erstellt. Dies wiederum beschreibt den Längen- und Breitengrad der TMC-Meldung. Da die Projektion von TMC „EPSG:4326“ ist, beim Google Web Toolkit aber die Google Projektion genutzt wird, müssen die Koordinaten auf „EPSG:900913“ transformiert werden, was die Methode „transform“ übernimmt. Anschließend können die Meldungen dem Layer hinzugefügt werden, indem an der Position der TMC-Meldung ein neuer Marker erstellt wird.

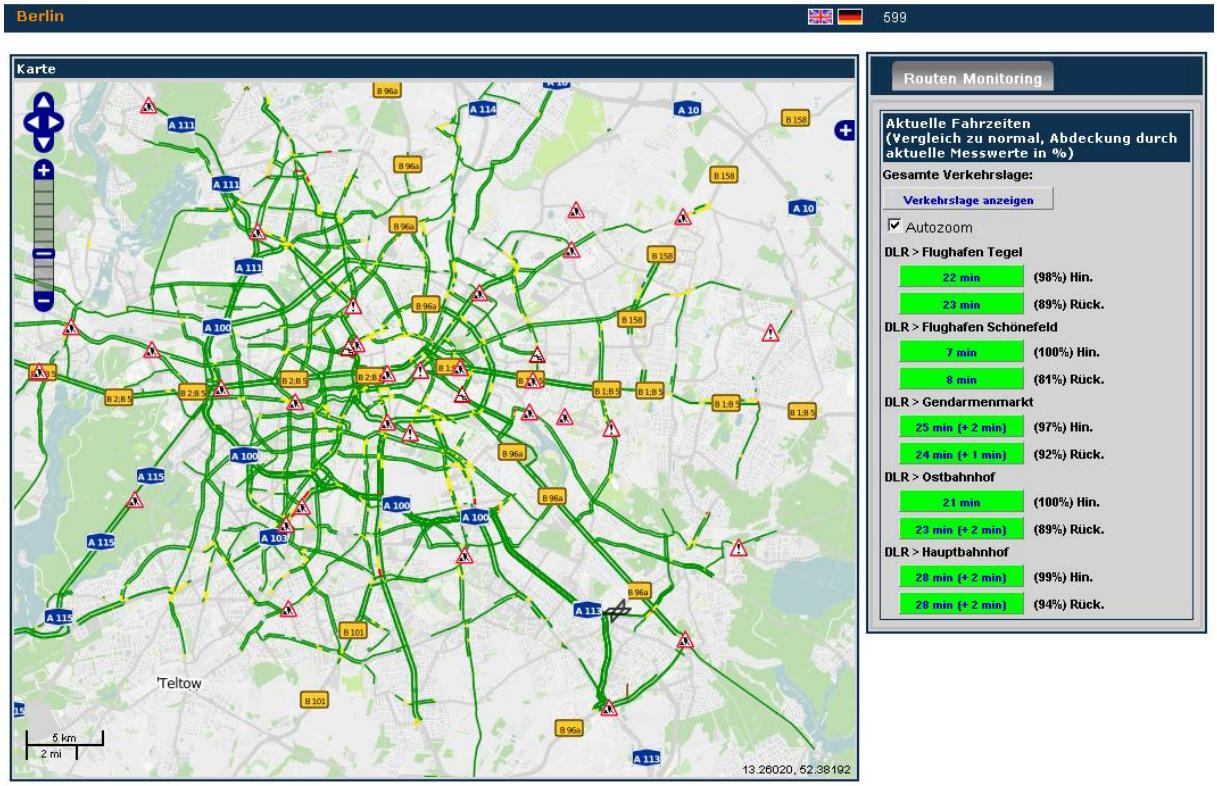


Abbildung 73: TMC-Marker

Die Abbildung 73 zeigt den TMC-Layer für die Stadt Berlin und das fertige Produkt. Wie zu sehen ist besitzt jeder Marker ein Warnsymbol, welches die TMC-Meldung kennzeichnet.

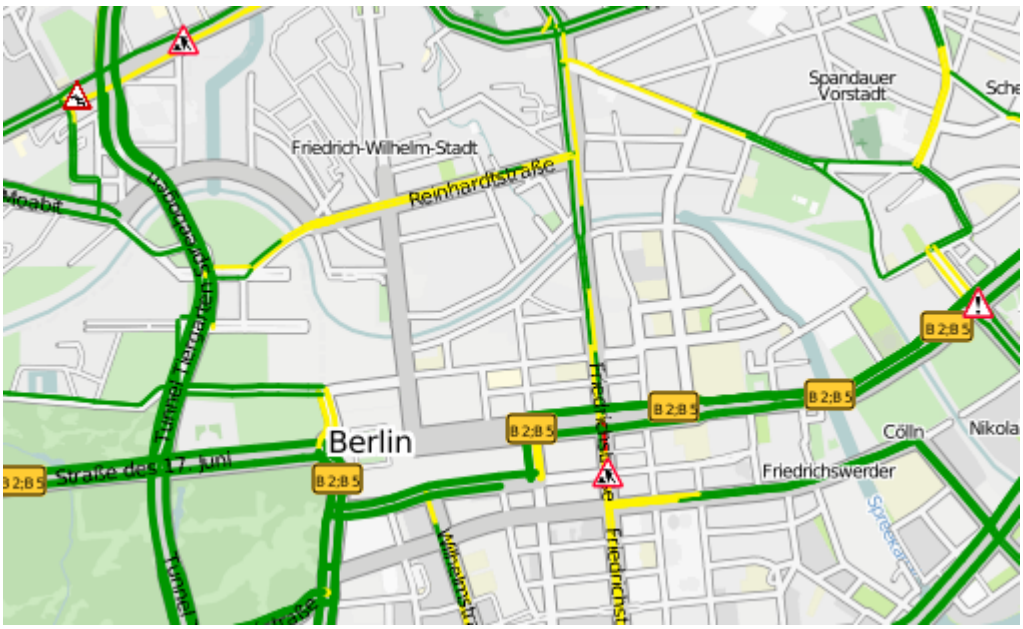


Abbildung 74: TMC-Marker Detail

Bei genauer Betrachtung, wie in Abbildung 74, zeigt sich, dass die Symbole unterschiedlich sind. Dies ermöglicht dem Betrachter auf den ersten Blick das TMC-Ereignis in die Kategorien einzuordnen:

- Achtung:  Diese TMC-Meldung kann keinem der folgenden Kategoerien zugeordnet werden.
- Baustelle:  Dieses Symbol kennzeichnet eine Baustelle.
- Rutschgefahr:  Dieses Symbol kennzeichnet eine Fahrbahn, auf der mit verringerter Reibung zu rechnen ist.
- Stau:  Dieses Symbol zeigt einen Stau an.
- Glätte:  Dieses Symbol zeigt vor allem im Winter glatte Fahrbahnen an.

Für weitere Informationen zu einer individuellen TMC-Meldung ist es möglich, sich alle Informationen aus dem Antwort-String des Servers anzeigen zu lassen. Dazu muss über die entsprechende TMC-Meldung mit der Maus gefahren werden, um das zuvor registrierte Ereignis „mouseover“ auszulösen.

Abbildung 75: TMC-Mouseover

Die Abbildung 75 zeigt, dass beim Bewegen der Maus über ein Symbol etwas in der Anwendung geschieht. Es wird ein neues Fenster erstellt, ein sogenanntes Popup-Fenster, das

ganz oben die Quelle, also den Radiosender bzw. dessen Frequenz, zeigt. Darauf folgt die Position in Länge und Breite. Durch eine Zeile abgesetzt wird die Decodierung der TMC-Meldung gezeigt, zuerst die Richtung, dann das Ereignis und den Empfangszeitpunkt sowie die Gültigkeitsdauer. Durch die registrierten Ereignisse „mousemove“ und „mouseoute“ wird das Fenster aus der Anwendung wieder entfernt, wenn die Mause bewegt wird oder sich von dem Symbol entfernt.

Die Darstellung der TMC-Meldungen benötigt bei einer etwaigen Veränderung der TMC-Elemente keinerlei Anpassung, sofern der gezeigte Codierungsrahmen eingehalten wird. Wenn z.B. mehr Informationen zu einer Meldung angezeigt werden sollen, muss die neue Beschreibung mit einem Semikolon getrennt an die jeweilige TMC-Meldung gehangen werden und wird anschließend ohne Veränderung der Cityrouteranwendung als neue Zeile im Popup-Fenster angezeigt. Lediglich bei einer Veränderung des Designs muss das Web Frontend angepasst werden.

Zur Ansicht des Ergebnisses kann unter der folgenden URL zu jeder Zeit die neue Verkehrslage betrachtete werden:

„http://www.cityrouter.de/traffic/berlin.html“

9. Webservice für neue Datenbasis

Für die Bereitstellung der TMC-Meldungen soll ein neuer Webservice entworfen werden, der unabhängig vom Cityrouter die neuen TMC-Meldungen bereitstellt. Für die Erstellung eines Webservices bieten sich zwei Lösungskonzepte an: das Simple Object Access Protocol“ (SOAP) oder der „REpresentational State Transfer“, kurz REST. Bevor die Umsetzung des Webservices beschrieben werden kann, soll im Folgendem die Vor- und Nachteile beider Konzepte verglichen werden. Dazu werden die Grundlagen beider Verfahren kurz beschrieben, ohne dabei zu tief den Aufbau zu erklären.

9.1. Grundlagen SOAP

Das “Simple Object Access Protocol“ beschreibt ein Verfahren, mit dessen Hilfe XML-basierende Nachrichten über ein Netzwerk ausgetauscht werden können. Allgemein beschreibt das World Wide Web Consortium[49] (W3C) einen SOAP-Webservice als einen Dialog von Maschine zu Maschine, der seine Interaktion auf ein Netz stützt. Dabei wird in einem festen spezifischen Format (WSDL)[50] kommuniziert. Die Betrachtung sieht dabei vor, dass eine Maschine etwas zur Verfügung stellt und andere Maschinen diese Informationen bzw. Daten nutzen. [56] Diese Beschreibung sieht folgende Komponenten zur Interkation vor:

- Dienstanbieter (Service Provider)
  - Bietet einen Dienst bzw. Service über das Web an.
  - Ferner annonciert und beschreibt der Anbieter den Service (WSDL)
  - Eine weitere Aufgabe ist die Implementierung, der Betrieb und die Wartung.
- Dienstanwender (Service Requestor)
  - Dieser nutzt das Angebot und sucht Dienste nach speziellen Kriterien (UDDI)[51].
  - Ferner ist es möglich, dass der Anwender die Dienste in eigene Programme integriert.
- Dienstmakler (Service Broker)
  - Dieser bietet die Möglichkeit Servicebeschreibungen zu speichern.
  - Ferner werden die Beschreibungen verwaltet und erleichtern das Auffinden von Diensten.

[49] Das W3C ist verantwortlich für die Standardisierung von Techniken, die das World Wide Web (WWW) betreffen, wie z.B. HTML, XHTML, XML.
[50] Die Web Service Description Language ist eine Protokoll unabhängige Beschreibungssprache für Netzwerkdienste.
[51] Universal Description, Discovery and Integration (UDDI), beschreibt einen standardisierten Verzeichnisdienst.

- Ist optional und wird oft nicht verwendet.

Diese Komponenten werden so in der Empfehlung vom W3C für SOAP-Webservices beschrieben. [56] Die sich daraus ergebene Abbildung ist in der Quelle nicht zu finden, lässt sich aber aus dem Text ableiten.

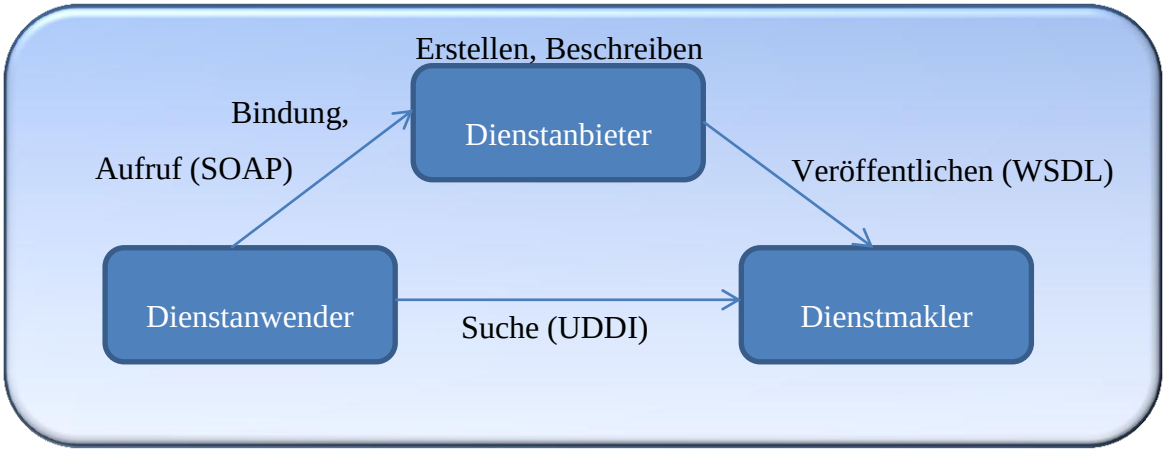


Abbildung 76: Architektur SOAP-WebService

Die Abbildung 76 „Architektur SOAP-WebService“ zeigt die grundlegende Architektur eines solchen Webservices, wie es im W3C beschrieben wird. Der Dienstanbieter erstellt hierzu einen Dienst und meldet diesen beim Dienstmakler an. Für die Veröffentlichung wird die Webservice Description Language genutzt, die nach W3C folgende 4 Komponenten enthalten muss: [57]

1. Adresse des Services
2. Interface aller Funktionen
3. Binding (Transport Protokoll)
4. Datentypen

Diese 4 Beschreibungsmerkmale müssen erfüllt sein, um den Service zu veröffentlichen. Anschließend kann über den Verzeichnisdienst „Universal Description, Discovery and Integration“ (UDDI) der Dienstanwender Services suchen. Diese Verzeichnisstruktur geht auf ein Microsoft Projekt zurück und beschreibt selbst wieder einen Dienst, der Dienste suchen kann. Da der Dienstmakler eine optionale Möglichkeit der Verbindung darstellt, ist die direkte Kommunikation über das SOA-Protokoll das am häufigsten genutzte Verfahren. Das „Simple Object Access Protocol“ (SOAP) ist ein plattformunabhängiges, XML-basiertes Protokoll, das es ermöglicht, Datenobjekte auszutauschen. Das Protokoll selbst ist unabhängig von den

Transportprotokollen und kann somit HTTP, FTP[52] bis hin zu Telnet[53] zum Aufbau der Übertragung nutzen. Dabei legt SOAP, vereinfacht beschrieben, fest, wie ein komplexes XML-Objekt, z.B. in HTTP verpackt und versendet werden soll. Diese Beschreibung soll nicht detailliert dargelegt werden, da lediglich die Grundlagen für SOAP-Webservices in diesem kurzem Abschnitt gezeigt werden sollen.

Wichtig ist, dass erkannt wird, dass dieses Verfahren eine Session, also eine feste Verbindung, benötigt. Diese grundlegende Kommunikationsweise zwischen Dienstanbieter und Dienstanwender beschreibt die Sicht eines SOAP-Webservices und wird in der Literatur als ein wesentliches und sehr verbreitetes Lösungskonzept für Webservices vorgestellt. [58] [59] [60] [56]

Die für dieses Projekt genutzte Architektur vereinfacht die Erstellung eines Webservices und nutzt nicht das beschriebene Verfahren, sondern einen einfachen „REpresentational State Transfer“ kurz REST. Diese Architektur soll den Umgang mit Diensten mit Netz vereinfachen. Im folgenden Abschnitt sollen auf die grundlegenden Ideen von REST und dessen theoretische Umsetzung eingegangen werden.

9.2. Grundlagen REST

Der „REpresentational State Transfer“ (REST) beschreibt ein Verfahren, dessen Schwerpunkt auf die Interkation mit zustandsbehafteten Ressourcen zurückzuführen ist. Grundsätzlich zeichnen REST 4 Eigenschaften aus, die bereits in der Dissertation „Architectural Styles and the Design of Network-based Software Architectures“ von Roy Fielding, dem Urvater der Architektur beschrieben wurden. Diese Eigenschaften sollen folgend kurz erklärt werden. [61]

Die zustandslose Kommunikation

Der Anspruch von REST ist die Erstellung einer zustandslosen Client/Server-Kommunikation, die über eine eineindeutig referenzierbare Ressource bestimmbar ist. Dabei wurde das Kernprinzip des HTT-Protokolls abstrahiert. Die Kommunikation erfolgt grundsätzlich zustandslos. Das bedeutet, dass der Client sich bei einer Anfrage nicht darauf verlassen kann, dass ein früherer erstellter Kontext immer noch vorhanden ist. Die Konsequenz ist, dass alle Informationen, die der Server zur Beantwortung der Anfrage benötigt in jeder Anfrage mitgesendet werden müssen. Der Vorteil bei solch einem Verfahren

[52] File Transfer Protokoll (FTP) Daten Übertragungsprotokoll
[53] **Tel**ecommunication **Net**work (Telnet) beschreibt den zeichenorientierten Datenaustausch zwischen Server und Client

ist, dass der Server nicht für eine etwaige laufende Interaktion begrenzte Systemressourcen, wie z.B. Arbeitsspeicher, zur Verfügung stellen muss. Daraus ergibt sich ein besseres Verhalten bei der Skalierbarkeit, Lastverteilung und Ausfallsicherheit. [61]

Eindeutige Ressource

Das zweite Prinzip von REST beschreibt eine eineindeutige Ressource. Der Begriff Ressource steht in diesem Zusammenhang für einen allgemeinen Oberbegriff. In dem konkreten Anwendungsfall wird die Ressource definiert und kann die Daten auf Wunsch in unterschiedlichen Formaten liefern, z.B. HTML, XML oder JSON bzw. als Grafik oder PDF. Das System verdank seinen Namen „REpresentational State Transfer“ dieser Möglichkeit. Jede Ressource ist dabei über eine eineindeutige Uniform Resources Identifier der URI referenziert. Bei einer gleichen Anfrage an eine konkrete Zustandsressource mit den gleichen Parametern wird der Server die gleiche Antwort geben. Dieses Verhalten ermöglicht außerdem die Herstellung eines immer gleichen Zustandes beim Client, wenn dies erwünscht sein sollte. [61]

Uniforme Schnittstellen

Die Uniform Schnittstelle beschreibt eine einheitliche Schnittstelle, die nur Operationen beinhaltet, die für alle Ressourcen gültig bzw. anwendbar sind. Dabei leiten sich einige aus der HTML-Welt ab, wie GET oder POST, andere aus der SQL-Welt, wie DELETE. Die jeweilige Operation entspricht dabei seiner ursprünglichen Entsprechung. Das bedeutet, dass GET immer nur Daten bekommt und nicht die Manipulation der Daten bewirken kann. Der Vorteil dieser Schnittstelle ist, dass sie universell ist und eine große Anzahl von Anwendungsfällen abdeckt. [61]

Hypermedia

Das Hypermedia Prinzip beschreibt ein Modell, dass vom Web-Browser bekannt ist. Dabei werden vom Server nicht nur Daten von den jeweiligen Ressourcen übertragen, sondern auch Zustandsübergänge vom Client in Form von Links[54]. Der Server liefert alle Informationen die vom Client benötigt werden, um Aktionen aufzuführen oder seinen Status zu aktualisieren. [61]

Diese kurze Einführung in REST soll reichen, um die beiden Systeme REST und SOAP vergleichen zu können.

[54] Ein Link (engl. Verknüpfung) ist eine eingebettete Verknüpfung, die zu einem anderen oder dem gleichen Dokument verweist.

9.3. SOAP vs. REST

Die direkte Gegenüberstellung von SOAP und REST soll den wesentlichen Vor- bzw. Nachteile der einzelnen Architekturen nochmals hervorheben und so beschreiben, welche Umsetzung sich für diese Arbeit am besten eignet.

Grundsätzlich ist REST einfacher zu entwickeln. Es benötigt eine Schnittstelle, die alle Operationen für die Ressource abdeckt. Bei SOAP können mehrere Schnittstellen individuell angepasst und erstellt werden. Bei genauerer Betrachtung zeigt sich, dass SOAP und REST keine wirklichen Konkurrenten sind. Vielmehr sollten dem Nutzer die Vor- und Nachteile der jeweiligen Systeme bekannt sein und der daraus resultierende Anwendungsfall abgeleitet werden. So ist es bei REST, auf Grund der fehlenden Session, nicht möglich, eine Geschäftslogik, wie eine Geldtransaktion beispielsweise, im vollen Maße abzubilden, da eine Überweisung nicht mehr zurückgezogen werden kann. Bei SOAP hingegen wird eine Session zwischen Server und Client aufgebaut, die die Implementierung solch einer Logik ermöglicht. Der Hauptunterschied der beiden Architekturen bezieht sich auf die Art der Verbindung. Somit kann man zusammenfassend sagen:

Das bei der Abbildung von komplizierten bzw. hochwertigen Logiken, bei denen die Nachvollziehbarkeit des Client gegeben sein muss SOAP das Mittel der Wahl sein sollte. Doch wenn es sich um eine einfache Schnittstellenarchitektur handelt, die, wie in diesem Fall, Daten zur Verfügung stellt, ohne die Notwendigkeit zu erachten, eine komplexe Kommunikation einzugehen, dann sollte die REST-Architektur verwendet werden. Diese hier getroffene Verallgemeinerung kann für jedes Projekt genutzt werden. Demnach ist zu entscheiden, ob eine Session benötigt wird und wenn ja, kann man diese mit REST (Hypermedia) abbilden oder nicht. Beziehungsweise sollte REST auf Grund seiner einfacheren Implementierung immer verwendet werden, wenn keine feste Zuordnung zwischen Client und Server zwingend erforderlich ist. Im Folgenden soll aus diesem Grund ein Webservice auf Basis der REST-Architektur entworfen werden, der die aktuellen TMC-Meldungen zur Verfügung stellt.

9.4. Umsetzung

Für die Umsetzung von REST wurde das Framework RESTLet genutzt. Diese Erweiterung für Java ermöglicht die Entwicklung eines Webservices in der „REpresentational State Transfer“ Beschreibung. Die notwendige Bibliothek ist die „org.restlet.jar“, welche von der offiziellen Seite heruntergeladen werden kann. [62] Ferner ist hier die Dokumentation zu

finden sowie detaillierte Beispiele, die einen guten Einstieg in die Programmierung ermöglichen.

```
/**
 * RESTLet Implementation of TMC-Message from Table tmc_active.
 * @author susk_da
 */
public class TMCService extends ServerResource {

    public static void main(String[] args) throws Exception {
        // Create a new Restlet component and add a HTTP server connector to it
        Component component = new Component();
        component.getServers().add(Protocol.HTTP, 8182);

        // Then attach it to the local host
        component.getDefaultHost().attach("/tmc/webservice/json", TMCService.class);

        // Now, let's start the component!
        // Note that the HTTP server connector is also automatically started.
        component.start();
    }

    @SuppressWarnings("unchecked")
    @Get
    public String toString() {

        try {
            ArrayList<TMCData> tmcArrayList =
                Connector.getInstance().getSelectAll("tmc_active");
            String result = "";
```

Abbildung 77: RESTLet

Die Abbildung 77 zeigt den wesentlichen Teil des neuen Webservices. Hierzu wird die Klasse „TMCServices“ erstellt, die von der Klasse „ServerResource“ erbt. Somit erhält der neue Service alle Eigenschaften von „ServerResource“. Diese repräsentiert eine eineindeutige Ressource, die über eine Identifikation erreichbar ist. Doch bevor der neuen Ressource eine URI gegeben werden kann, muss eine Objekt der Klasse „Component“ erstellt werden. Dieses Objekt kontrolliert die Verbindung bzw. ermöglicht die Erstellung der selbigen, in dem dem Server (component.getServers()) das Kommunikationsprotokoll und der zu verwendende Port (.add(Protocol.HTTP, 8182)) übergeben wird.

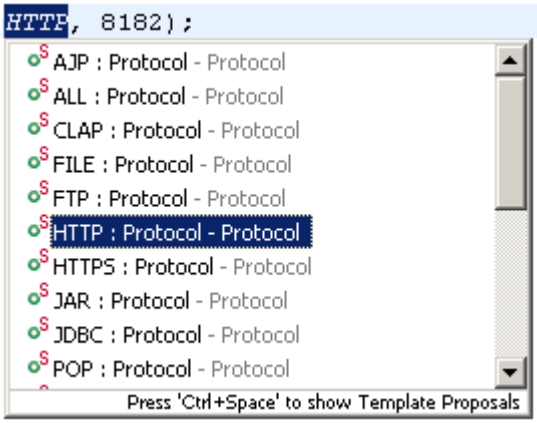


Abbildung 78: RESTLet Protokolle

Möglich sind auch andere Protokolle, die von der API unterstützt werden, wie die Abbildung 78 zeigt. So wird beispielsweise auch eine HTTPS-Verbindung angeboten, die eine abhörsichere Übertragung ermöglicht. Da der Dienst jedem zur Verfügung stehen soll, der ihn nutzen möchte, ist eine HTTP-Verbindung für eine Übertragung ausreichend. Nach dem die Parameter für die Verbindung dem Objekt der Klasse „Component“ zugeordnet wurden, kann die Identifikation (URI) dem Host, also dem Dienstanbieter, zugeordnet werden. Dazu liefert wieder „Component“ das notwendige Objekt (component.getDefaultHost()), den Host, an den die Adresse und die zugehörige Ressource gehängt wird (.attach("/tmc/webservice/json", TMCService.class)). Die Ressource ist die Klasse selbst, da sie durch das Erben von „ServerResource“ eine Ressource ist. Anschließend kann die Verbindung mit „component.start()“ gestartet werden. Was die Ressource dem Client zurückgeben soll, kann mit den entsprechenden Annotationen beschrieben werden. Eine Annotation in Java beginnt mit einem „@“ Zeichen, wie „@Get“ aus Abbildung 77 diese Metadaten im Quelltext werden genutzt, um zusätzliche Informationen zu Programm und Bytecode zur Verfügung zu stellen. Diese Herangehensweise wurde erstmals in der JSR (Java Specification Request) 175 beschrieben. [63] Die „@Get“ Annotation über der Methode „toString“ beschreibt das Verhalten der Ressource bei einer GET-Anfrage eines Nutzers. Bei Aufruf der Ressource wird die Tabelle „tmc_active“ in die Liste „tmcArrayList“ ausgelesen, so dass alle TMC-Meldungen als Objekte vom Typ „TMCData“ in der Liste vorliegen. Die Verfahrensweise entspricht der Erstellung der Daten aus Kapitel 8. Abschnitt 2. Auch hier werden mit JDBC die Daten aus der Datenbank entnommen und in einem Objekt der Klasse „TMCData“ gespeichert.

Nachdem die TMCData Objekte erstellt wurden, werden diese im JSON-Format ausgegeben. Die „Java Script Object Notation“ ist ein kompaktes Datenformat, das sowohl von einem Menschen als auch von Rechner einfach und schnell gelesen werden kann.

```
TMCMessage(Json): {"bNew":"1","type":"New",
TMCMessage(Json): {"bNew":"1","type":"New",
TMCMessage(Json): {"bNew":"1","type":"New",
TMCMessage(Json): {"bNew":"1","type":"New",
TMCMessage(Json): {"bNew":"1","type":"New",
TMCMessage(Json): {"bNew":"1","type":"New",
```

Abbildung 79: Webservice Ausgabe

Die Abbildung 79 zeigt einen Ausschnitt der Ausgabe, die im Browser beim Nutzer zu sehen ist. Jede Zeile der Anzeige entspricht einer TMC-Meldung aus der Tabelle „tmc_active“. Das JSON-Format für solch eine Meldung beginnt mit einer geschweiften Klammer „{„ und endet mit „}“. Jedes JSON-Element besteht aus zwei Strings, dem Bezeichner, z.B. „bNew“ und dem Wert, z.B. „1“. Diese werden mit einem Doppelpunkt „:“ getrennt. Die Elemente selbst werden mit einem Komma „ , “ getrennt. Wie die Abbildung andeutet, entsprechen die Einzelelemente dem Format von „TMC_USERDATA“, wie es unter anderem in Kapitel 5. Abschnitt 2.4. beschrieben wird.

Dieses einfache Verfahren ermöglicht die Bereitstellung der validierten TMC-Meldungen für einen beliebigen Nutzer. Für die Erreichbarkeit des Services im Internet muss die Anwendung als Dienst auf einem zugänglichen Internetrechner registriert und gestartet werden. Dies soll im folgenden Kapitel beschrieben werden.

Zur Ansicht des Ergebnisses kann unter der folgenden URL zu jeder Zeit eine Anfrage an den Webservice gestellt werden:

„http://129.247.218.175:8080/tmcwebservice/json“

10. Testphase

In diesem Kapitel wird beschrieben, wie die zuvor erstellten Anwendungen Datenimport, „TMC-Process Unit“ und der Webservice auf einem Server als Dienst registriert und gestartet werden. Ferner wird beschrieben, wie sich die Services im Produktivsystem verhalten und ob erwarte Reaktionen bei bestimmten Situationen eintreten oder nicht.

10.1. Inbetriebnahme

Die Inbetriebnahme der Anwendungen als Services lässt sich in drei Teile gruppieren, den „Datenimport“, die “TMC Process Unit“ und dem Webservices. Jede der Anwendungen soll auf dem „Ikarus“ Server des DLR als Service in Betrieb genommen werden. Dieser Windows NT 2003 Server läuft mit dem Service Pack 2 und der Standard 64-Bit Edition.

10.1.1. Datenimport

Die Anwendung „GPSTMCReceiver“ wurde in C++ geschrieben, siehe Kapitel 5. Abschnitt 1. Um diese als Service auf dem Server zu registrieren, wird eine ausführbare Datei benötigt, die in den folgenden 5 Schritten erzeugt werden kann:

1. **Präprozessor**
   Der Präprozessor wertet die Anweisung für den Prozessor aus, wie den Befehl „#include“, um beispielsweise Header-Dateien der Klasse hinzuzufügen. Diese ausgewerteten Informationen werden dem C++Compiler übergeben.

2. **C++Compiler**
   Der C++Compiler erzeugt aus allen Quelltextdateien C-Code. Dieser Umweg über die Sprache C ist historisch bedingt. Da C++ aus C entstanden ist, haben sich die Compiler dem aufbauendem System der Sprachen bedient und bauen so selbst aufeinander auf.

3. **C-Compiler**
   Die Sprache C ihrerseits ist aus der Sprache Assembler hervorgegangen. Somit verwundert es nicht, dass der Compiler den C-Code in eine Assemblerdatei übersetzt.

4. **Assembler**
   Assembler selbst erzeugt aus der durch den C-Compiler gelieferten Datei eine

Objektdatei mit der Endung „.obj“. Diese Dateien sind vom Prozessor lesbar und enthalten die Maschinensprache sowie Informationen über den Code, z.B. die Positionen einer Variablen.

5. **Linker**

   Der Linker verbindet die einzelnen Objektdateien bzw. Code-Schnipsel zu einer ausführbaren Datei mit der Endung „.exe“. Diese Datei kann auf jedem Windowsbetriebssystem ausgeführt werden.

Diese Schritte müssen nicht alle einzeln ausgeführt werden, neue C++Compiler enthalten bereits den Präprozessor sowie die C-Übersetzung und den Assemblermechanismus zur Erstellung einer Objektdatei. Trotzdem ist es möglich Zwischenausgabeformate zu definieren. Des Weiteren ist es nicht nur möglich direkt Objekt-Dateien zu erstellen. Neue Entwicklungsumgebungen, wie das genutzte „Visual Studio 2010“, bieten die Integration aller notwendigen Schritte an. Somit ist es möglich, sofort die ausführbare Exe-Datei zu erstellen. Die so erstellte „GPSTMCReceiver.exe“ kann auf dem Server Windowsserver laufen.

Für die Registrierung eines neuen Dienstes auf dem Server stellt Windows NT zwei Dienstprogramme zur Verfügung. Der Dienst Instrsrv.exe ermöglicht die Installation und Deinstallation von Diensten in einer Windows NT Umgebung, und Srvany.exe ermöglicht das Ausführen beliebiger Windowsanwendungen als Dienste, wie die zuvor erstellte „GPSTMCReceiver.exe“.

Zuerst muss in der Windows-Registry ein neuer Parameter angelegt werden. Diese Datenbank verwaltet alle Programme, Dienste, Benutzerinformationen und deren Rechte. Ferner werden Eigenschaften und deren Bedeutung sowie Zusammenspiel im Betriebssystem gespeichert. Zur Erstellung eines neuen Dienstes muss ein neuer Service in dieser Datenbank mit entsprechenden Parametern angelegt werden. Der Aufruf der folgenden Zeile bewirkt, dass der Registry ein neuer Service mit dem Namen „dlr.tmc.receiver“ hinzugefügt und das ein Unterordner mit Namen „Parameters“ erstellt wird:

```
reg add HKLM\SYSTEM\ControlSet001\Services\DLR.tmc.receiver\Parameters
```

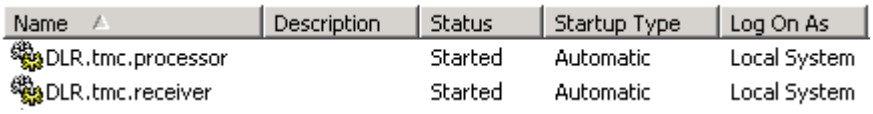
Anschließend müssen die Werte bzw. Parameter noch gesetzt werden. Dazu wird folgender Befehl ausgeführt:

reg add HKLM\SYSTEM\ControlSet001\Services\DLR.tmc.receiver\Parameters /v AppDirectory /t REG_SZ /d D:\Programme\DLR\TMC-Receiver

Dieser Aufruf fügt den Parametern die Variable für den Ort der Anwendung hinzu. Dieser wird unter „AppDirectory“ mit dem Pfad „D:\Prgramme\DLR\TMC-Receiver“ gespeichert. Die Anwendung selbst bekommt eine eigene Parameterzuweisung in Form einer Variablen, wie dieser Befehlt zeigt:

reg add HKLM\SYSTEM\ControlSet001\Services\DLR.tmc.receiver\Parameters /v Application /t REG_SZ /d GpsTmcReceiver.exe

Die Variable „Application“ enthält den String „GpsTmcreceiver.exe“, also den Namen der Anwendung selbst. Nach dem Ausführen dieser drei Befehle wurde ein neuer Dienst im System registriert und kann jederzeit gestartet werden. Hierzu kann über Nutzoberfläche „Services“ des Servers die vorhandene Funktionalität genutzt werden, da die Dienste jetzt bekannt sind.



Name	Description	Status	Startup Type	Log On As
DLR.tmc.processor		Started	Automatic	Local System
DLR.tmc.receiver		Started	Automatic	Local System

Abbildung 80: Dienste unter Windows-Services

Die Abbildung 80 zeigt die registrierten Dienste in der Windowsanwendung „Services“. Der in der Spalte „Name“ stehende Text entspricht dem Namen des Services, der im ersten Aufruf definiert wurde.

10.1.2. TMC Process Unit

Die „TMC Process Unit“ ist ein in Java geschriebenes Programm, siehe Kapitel 5. Abschnitt 2. Um diese als Service auf dem Server zu registrieren wird ebenfalls eine ausführbare Dateien in Form einer EXE-Datei benötigt, die in den folgenden 5 Schritten erzeugt werden kann:

1. **Compiler**

   Der Java-Compiler analysiert den geschrieben Code in den Dateien mit der Endung „.java“, den Quellcode. Die Analyse bezieht sich dabei auf mögliche Fehler in der Syntax, Zuweisungen oder Typumwandlung.

2. **Debugger**

   Der Debugger ist ein Teil des Compilers, der die eigentliche Fehleranalyse durchführt.

3. **Bytecode-Generator**

   Der Bytecodegenerator ist der zweite Teil des Compilers, der aus den geprüften Quellcodedateien eine maschinenlesbare Datei mit der Endung „.class“ erstellt. Die Besonderheit zu dessen Entsprechung aus der Sprache C++, der EXE-Datei, ist, dass der Bytecode plattformunabhängig interpretiert werden kann.

4. **Interpreter**

   Der letzte Schritt beinhaltet das Lesen des Bytecodes. Dazu interpretiert die „Java Virtuell Maschine“ die maschinenlesbaren Dateien und führt die entsprechenden Anweisungen auf einem beliebigen System aus.

5. **Java-Virtuelle-Maschine**

   Dieser virtuelle Prozessor übersetzt seinerseits den für ihn verständlichen Maschinencode in den für den laufenden Prozessor verständlichen Maschinecode. Die JVM fungiert somit als ein Übersetzer, der die Sprache Java unabhängig von der jeweiligen Plattform macht. Sofern die JVM auf einem Rechner installiert ist und diese unterstützt wird, kann das Java-Programm auf allen Rechnern ausgeführt werden.

Als Konsequenz wäre es möglich, das Java-Programm auf den Windows NT Server zu starten, wenn dieser eine JVM installiert hat. Der elegantere Weg ist es jedoch, eine EXE-Datei aus dem Programm zu erstellen. Dazu muss als fünfter Schritt aus dem Java-Programm mittels eines Wrappers[55] die EXE erstellt werden. Dazu wird ein Java-Archiv benötigt, das die Klassendateien bzw. die maschinenlesbaren Dateien beinhaltet. Dieses Archiv kann in der verwendeten Entwicklungsumgebung „Eclipse“ als Exportmöglichkeit angegeben werden und wird mittels einer Manifest-Datei erstellt. Diese Datei enthält alle Angaben zum Projekt, wie z.B. Version, Klassenpfad, den Ersteller, benötigte Klassen und die Hauptklasse, den Startpunkt des Projektes. Das so erstelle Archiv „TMCProcessor.jar“ wird mit dem EXE-Wrapper „Launch4J“ in eine EXE-Datei umgewandelt. Für diese Aufgabe stehen zahlreiche Anwendungen zu Verfügung, die nach eigenen Angaben alle das gewünschte Ergebnis

[55] Wrapper (Verpacker) beschreibt eine Software, die eine andere umhüllt. Der Zweck der Umhüllung ist es, die Softwarefunktionalität zu erhalten, wenn einem anderen System die Software zugänglich gemacht wird.

versprechen. Die Entscheidung für „Launch4J“ resultiert aus dem Entwicklerforum „Viralpatel“ [64], bei dem die Software als „Gut“ empfohlen wird. Die Bedienung ist über eine Nutzeroberfläche möglich und benötigt lediglich die Angabe des Pfades zum Java-Archiv und dem Ausgabepfad. Die erstellte EXE-Datei „TMCProcessor.exe“ kann jetzt als Service auf dem Server registriert werden. Die Vorgehensweise ist genau wie bei dem „Datenimport“, nur dass der Servicename in „DLR.tmc.processor“ zu ändern ist.

reg add HKLM\SYSTEM\ControlSet001\Services\DLR.tmc.processor\Parameters

Die Angabe der Variablen ist natürlich ebenfalls wie folgt anzupassen:

reg add HKLM\SYSTEM\ControlSet001\Services\DLR.tmc.processor\Parameters /v AppDirectory /t REG_SZ /d D:\Programme\DLR\TMCProcessor

Wie an dem oberen Befehl zu sehen ist, hat sich der Pfad zur EXE-Datei verändert sowie der Name der EXE-Datei selbst, wie folgender Befehl zeigt:

reg add HKLM\SYSTEM\ControlSet001\Services\DLR.tmc.receiver\Parameters /v Application /t REG_SZ /d TMCProcessor.exe

Die beiden Dienste sind jetzt auf dem Server angemeldet und können gestartet werden!

10.1.3. TMC Webservice

Der Webservice wurde ebenfalls in Java programmiert, siehe Kapitel 9. Abschnitt 4. Die Herangehensweise ist somit bis zur Erstellung der EXE-Datei exakt die gleiche, wie im Abschnitt 10.1.2. Die Registrierung des Services benötigt wiederum eine kleine Anpassung der entsprechenden Befehle. Der Name des Services ist wieder individuell:

reg add HKLM\SYSTEM\ControlSet001\Services\DLR.tmc.webservice\Parameters

Der bekannte Befehl erzeugt in Registry den neuen Service „DLR.tmc.webservice“.

reg add HKLM\SYSTEM\ControlSet001\Services\DLR.tmc.processor\Parameters /v AppDirectory /t REG_SZ /d D:\Programme\DLR\TMCWebservice

Der Pfad zum Service wird entsprechend der Ablage der EXE-Datei auf dem Server angepasst.

reg add HKLM\SYSTEM\ControlSet001\Services\DLR.tmc.receiver\Parameters /v Application /t REG_SZ /d TMCWebservice.exe

Zuletzt wird der Name der EXE-Datei als Variable in der Registry vermerkt.

Die TMC-Kette vom Datenimport über dessen Prozessierung bis hin zur Bereitstellung der TMC-Daten als Webservice kann jetzt als Dienst auf dem Server gestartet werden.

10.2. Lauffähigkeit

Die Lauffähigkeit der einzelnen Module ist unterschiedlich zu bestimmen, da der „Datenimporter“ keine Abhängigkeiten zu anderen Programmen besitzt, die „TMC Process Unit“ und der „TMC-Webservice“ schon.

10.2.1. Datenimporter

Der „Dateimporter“ ist ein eigenständiger Service der über eine Antenne, dem TMC-Empfänger, die Daten erhält und decodiert. Er besitzt keine Abhängigkeiten zu anderen Diensten und stellt den Anfang der TMC-Verarbeitungskette dar. Demnach läuft der Service so lange, wie der Server, auf dem er läuft, in Betrieb ist. Da die TMC-Meldungen nach ISO-Standard 14819 definiert sind, ist kein Fehler beim Parsen der Meldung zu erwarten. Somit ist die Lauffähigkeit dieses Moduls gesichert und soll sich im Langzeittest bewähren.

10.2.2. TMC Process Unit

Die „TMC Process Unit“ ist vom „Datenimporter“ abhängig, der die TMC-Meldungen bereitstellt. Bei einem Ausfall des Importservices würde die TMC-Verarbeitungskette nicht mehr funktionieren, da die elementarste Information fehlt: die Meldung selbst. In diesem Fall wartet der Service bis eine neue Meldung im System vorliegt. Problematischer wird der Ausfall einer der Services, die zum Abgleich von TMC mit FCD verwendet wurden, den „Routingservice“ und den „TraveltimeService“. Bei einem Ausfall einer dieser Services funktioniert der Datenabgleich nicht mehr. Da diese nur verwendet werden wenn TMC-Meldungen eintreffen, die einen Geschwindigkeitsbezug haben bzw. einen Stau oder stockenden Verkehr suggerieren, muss der „TMCProcessor“, also die Einheit in der „TMC Process Unit“, die diese Verarbeitung vornimmt, weiter funktionieren. Bei einem Ausfall des Prozessors würde der gesamte Service nicht mehr ordnungsgemäß ablaufen. Die Sicherung dieser kritischen Bereiche übernimmt eine eigene Fehlerklasse innerhalb des Services. Dies

ist die Klasse „AddictedServiceException“, die von der Klasse „Exception“ erbt. Die entsprechenden Methoden innerhalb der Validierung, bei denen die genannten Services genutzt werden, werden mit einem „throw new AddictedServiceException()“ versehen.

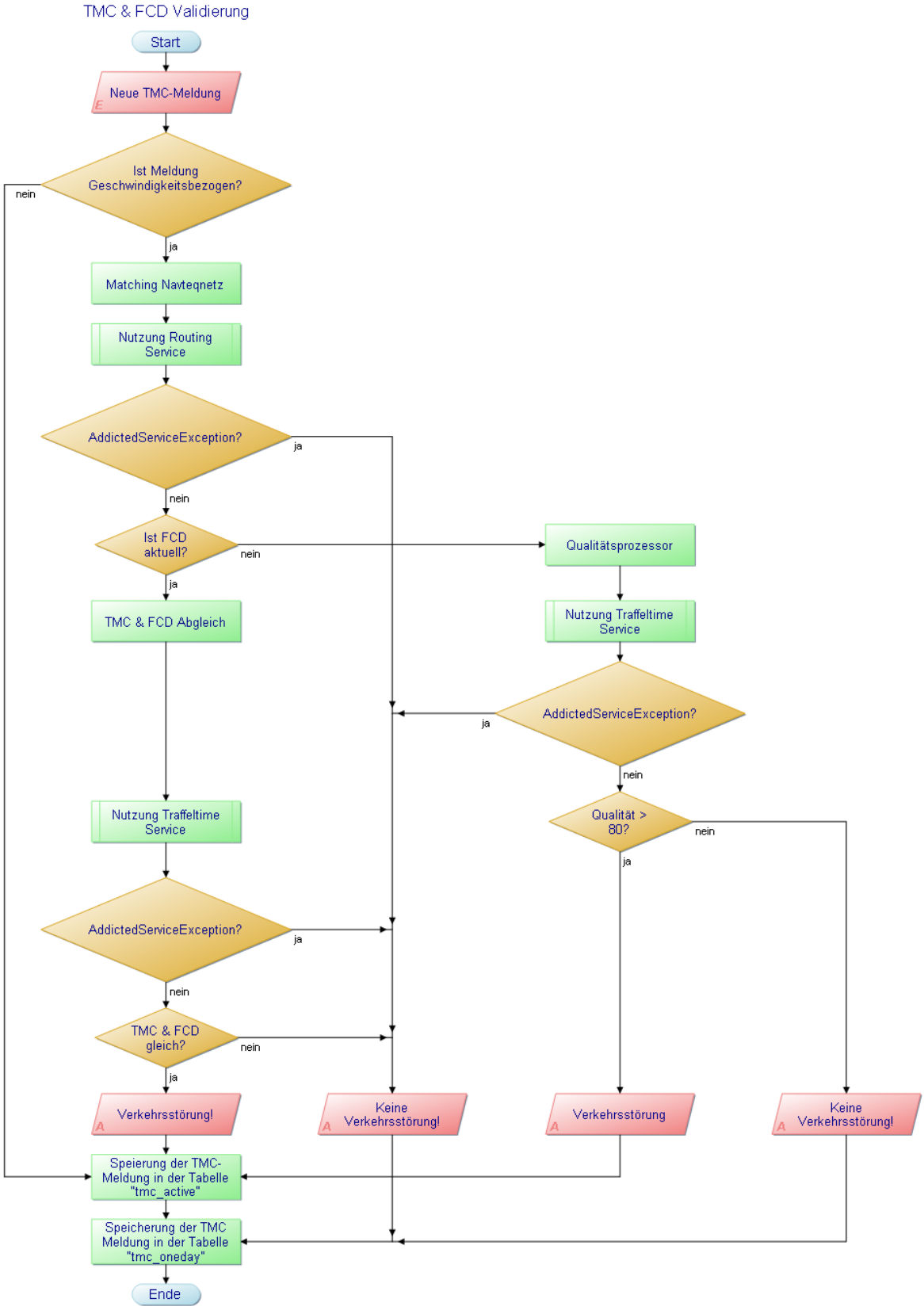


Abbildung 81: Prozessablaufplan TMC & FCD Validierung Exception

Diese Abbildung 81 entspricht der Abbildung 61 aus Kapitel 7. Abschnitt 2. „Prozessablaufplan“ und wurde um die Fehlerbehandlung erweitert. Beim Auftreten eines Fehlers im Abschnitt „Matching Navteqnetz“, „TMC & FCD Abgleich“ oder beim „Qualitätsprozessor“ wird die TMC-Meldung sofort in Tabelle „tmc_oneday“ geschrieben, nicht aber in „tmc_active“, damit die Meldung zu späteren Auswertung nicht verloren geht aber nicht in der aktuellen Verkehrslage angezeigt wird. Durch dieses Verfahren ist es möglich, dass die „TMC Process Unit“, wenn auch eingeschränkt, TMC-Meldungen verarbeitet und zur Anzeige bereitstellt bzw. in den Datenbanktabellen speichert. Ohne die vorhandenen Service zur Validierung der TMC-Daten werden nur noch nicht geschwindigkeitsbezogene Daten in der aktuellen Verkehrslage angezeigt.

10.3. Auswertung

Nach der Inbetriebnahme am ersten Oktober und Sicherung der Lauffähigkeit durch entsprechende Fehlerbehandlung wurden die Services 4 Wochen im Regelbetrieb bis zum 27. Oktober getestet. In diesem Zeitraum fiel der „Routing Service“ 3-mal aus, der „TravelTime Service“ 4-mal. Die erstellten Dienste konnten ihren Betrieb jedoch in eingeschränkter Form wie geplant fortsetzen. Das beschriebene Fehlermanagement konnte somit im Regelbetrieb seine Funktionalität erfolgreich unter Beweis stellen. Der „Datenimporter“ sowie die „TMC Process Unit“ und der „WebService“ liefen in dem gesamten Zeitraum durch, wie die auf der CD befindlichen Log-Dateien bestätigen.

```
2011-10-4 14:37:36,095 INFO  [Thread-3] main: PSM_TMC is reading not validating FCD "TravelTime Service" isn't available
2011-10-4 14:47:36,103 INFO  [Thread-3] main: PSM_TMC is reading
2011-10-4 14:57:36,110 INFO  [Thread-3] main: PSM_TMC is reading
2011-10-4 15:07:36,118 INFO  [Thread-3] main: PSM_TMC is reading not validating FCD "Routing Service" isn't available
```

Abbildung 82: TMC Log 2011-10-4

Die Abbildung 82 zeigt einen Ausschnitt aus der Log-Datei vom 4. Oktober 2011. Dieser Zeitabschnitt zeigt, dass um 14:37 Uhr TMC gelesen werden konnte. Dies wiederum bedeutet, dass der „Datenimporter“ funktionierte sowie die „TMC Process Unit“, die die Log-Datei schreibt. Ferner ist abzulesen, dass zu diesem Zeitpunkt kein Abgleichen zwischen FCD und TMC möglich war, da der „TavelTime Service“ nicht erreicht werden konnte. Die Log-Datei wird alle 10 Minuten von der „TMC Process Unit“ mit einem aktuellen Status gefüllt. Demnach war um 14:47 Uhr der „TravelTime Service“ wieder erreichbar. Der hier gewählte Ausschnitt aus der Log-Datei zeigt ebenfalls um 15:07 Uhr mit der Meldung: „PSM_TMC is

reading not validating FCD „Routing Service“ isn’t available“, die Ausgabe der „TMC Process Unit“, als der „Routing Service“ nicht erreichbar war. Die Ausgaberoutine, die diese Zeile erzeugt, wird von einem eigenen Thread übernommen (Thread 3 innerhalb der Anwendung), der alle eingehenden Fehlermeldungen der letzten 10 Minuten sammelt und diese in einer Zeile zusammenfasst. Wenn keine Fehler in den letzten 10 Minuten aufgetreten sind, wird in die Log-Datei die Meldung: „PSM_TMC is reading“ geschrieben. Dies bedeutet, dass TMC gelesen und ohne Störung verarbeitet werden kann.

Die Testphase wird zum 28. Oktober 2011 abgeschlossen, da das System robust zu seien scheint sowie die nötigen TMC-Daten liefert und mit FCD validiert. Des Weiteren konnten keine Fehler zur Laufzeit im Testzeitraum festgestellt werden, dies sichert eine etwaige Weiterverwendung oder kommerzielle Nutzung der neuen TMC-Verarbeitungskette und eröffnet mögliche Perspektiven. Welche Möglichkeiten sich aus dem neuen System perspektivisch ableiten lassen soll im Kapitel 11. beschrieben werden.

11. Resultate

Die Aufgabe die innerhalb dieser Arbeit gelöst werden sollte, war die Erstellung einer neuen Verkehrslage, durch die Zusammenführung von FCD und TMC. Wobei der Mehrwert aus der gestellten Aufgabe die Optimierung der Verkehrssituationsanalyse sein sollte, wie aus dem Titel zu interpretieren ist:

Validierung und Erweiterung von bestehenden „Floating Car Daten“ durch den „Traffic Message Channel“ Dienst des „Radio-Data-System“ zur Optimierung der Verkehrssituationsanalyse in Berlin.

Diese als Kausalitätskette zu verstehende Bedingung wird bereits in der Einleitung als elementare Aussage beschrieben. Demnach führt eine verbesserte Übersicht des aktuellen Verkehrsgeschehens in Form einer besseren bzw. erweiterten Verkehrslage zu einer besseren Analyse der selbigen. Innerhalb der Arbeit wird gezeigt, wie FC-Daten beim DLR verarbeitet und interpretiert werden und welche Möglichkeiten der Darstellung es für das FCD-System gibt. Ferner wird aufbauend auf der bestehenden Architektur ein eigenes System entworfen, das TMC-Meldungen in das bestehende System integriert und an den relevanten Punkten verknüpft. Dabei wurde auf eine hohe Modularität der neuen Verarbeitungskette und auf dessen Ausfallsicherheit bzw. Unabhängigkeit von den bestehenden Systemen geachtet. Die neu erstellte Verarbeitungskette, ist ein sehr wichtiges Resultat: Es erweitert das bestehende FCD-System bzw. die daraus resultierende Verkehrssituation und ermöglicht die Validierung der Daten.

Des Weiteren konnte gezeigt werden, wie TMC-Meldung über GNS 3.0 zu empfangen und zu decodieren sind und welche Qualitäts- und Quantitätstendenz die hiesigen TMC-Radiosender von Berlin haben. Dabei zeigt sich, dass der Radiosender „88.8 Radio Berlin“ nicht nur qualitativ sondern auch quantitativ der Sender ist, der sehr gute TMC-Meldungen im urbanen Bereich Berlin ausstrahlt. Diese neue Erkenntnis kann sich dabei nicht auf eine große Datenbasis stützen. Vielmehr beschreibt die vorliegende Erkenntnis eine Tendenz für einen bestimmten Zeitraum, die aus den erhobenen Daten abgeleitet werden konnte.

Ferner wird gezeigt, wie TMC-Meldungen zusammengefasst werden können, um eine übergeordnete Meldung zu erstellen. Diese Möglichkeit aus Kapitel 6. Abschnitt 4. soll im Ausblick noch einmal aufgegriffen werden.

Ein weiteres Resultat dieser Arbeit ist die Gegenüberstellung von FCD und TMC aus Kapitel 6. Abschnitt 5., die zeigt, dass TMC gegenüber FCD eine gewisse Trägheit zu zeigen scheint. Der Zeitversatz von einer Stunde ergibt sich bei der Betrachtung aus der Aufnahmemöglichkeit der einzelnen Systeme. Demnach erfasst das FCD-System eine Verkehrsstörung in dem Moment, wenn ein Taxi den entsprechenden Abschnitt befährt. TMC ist keine Verkehrserfassungssystem sondern ein Verkehrslageübertragungssystem. Dies führt dazu, dass erst eins der angeschlossen Erfassungssysteme, wie Polizei oder Radiohörer, die Meldung dem jeweiligen Sender vermitteln müssen.

Das wichtigste Resultat dieser Arbeit ist der Mehrwert für die Verkehrssituationsanalyse, wie es auch im Thema als wesentlicher Punkt zu finden ist. Mit Hilfe der neuen Datenbasis ist es erstmalig möglich, Verkehrsstörungen mit einem Ereignis zu verbinden. Die ursprüngliche Datenbasis gab zwar Auskunft über die Verkehrssituation, aber nicht über den Grund.

Abbildung 83: FCD & TMC

Die Abbildung 83 zeigt den aus Kapitel 7. Anschnitt 3. bekannten Informationsgehalt, der die neuen Daten als Synergieeffekt in einer Darstellung vereint. Die FC-Daten zeigen auf der Friedrichstraße eine Beeinträchtigung der Geschwindigkeit von mindestens 50 Prozent und Maximal 75 Prozent (gelbe Linie). Diese Information ist ausreichend, um bei einem Routing die Straße zu vermeiden und bei einer Verkehrssituationsanalyse die

Geschwindigkeitsinformation zu berücksichtigen, nicht aber den Grund der Beeinträchtigung zu erklären. Mit Hilfe der TMC-Meldung kommt in jedem Fall die neue Kenngröße des Ereignisses hinzu, sofern überhaupt TMC-Meldungen für diese Position vorliegen. In dem Popupfenster in Abbildung 83 wird ersichtlich, dass sich auf diesem Straßenabschnitt eine Baustelle befindet, die für die aktuelle Verkehrssituation verantwortlich sein könnte, nicht muss, da ebenfalls ein Unfall oder Personen auf der Fahrbahn den Verkehr beinträchtigen könnten. Unter Vernachlässigung des unbestimmten Fehlers bei der Berücksichtigung zweier kohärenter Daten für eine Position, kann der Informationsgehalt für die Verkehrssituationsanalyse um den Faktor 2 erhöht werden. Möglich ist aber auch eine weitere Steigerung, um den Faktor 3 oder 4 und mehr. Die Begrenzung ist lediglich an TMC selbst gebunden, die nach ISO 14819-1 Standard 5 Zusatzinformationen tragen kann.

12. Ausblick

In diesem Kapitel sollen noch einmal einige der beschriebenen Resultate perspektivisch aufgegriffen werden, um weitere Möglichkeiten und zukünftige Arbeiten zu diesem Thema zu motivieren. Bevor diese Betrachtung beschrieben wird, soll ein Einblick in mögliche Alternativen einen Ausblick in die Haltwertzeit von TMC ermöglichen.

Auf die Fragestellung, nach Alternativen zu TMC, ist jedes System zu nennen, das ortsbezogene Ereignisse für den Straßenverkehr erstellt, wie die Verkehrslage von „Verkehr24.de“ [65] oder die vom „ADAC“ [66], die auf der Basis von eigenen Systemen Staumeldungen bzw. Verkehrsstörungen erstellen. Alle diese Dienste sind über ihre Webseiten erreichbar und geben den Nutzern vergleichbare Informationen wie TMC. Der Zugang wird dem Nutzer erschwert, da ein Internetanschluss nötig ist, um Dienste dieser Art nutzen zu können. Der Fahrer müsste also während der Fahrt mit seinem Smart-Phone oder einem anderen internetfähigen Gerät auf die entsprechenden Services zugreifen und in regelmäßigen Abständen auf das Handy schauen. Vielleicht ermöglicht in naher Zukunft eine Handyanwendung die komfortablere Nutzung dieser Dienste bzw. führt diese alle zusammen und stellt die Verkehrslage in geeigneter Form dar. Bis jetzt wurde dies allerdings noch nicht realisiert und in Anbetracht der Schnittstellenproblematik und dessen Uneinheitlichkeit ist wohl in naher Zukunft mit solch einer Anwendung auch nicht zu rechnen. [67] So die Quintessenz des Treffens des TMC-Forums in Pretoria (Südafriak). Des Weiteren ist in 90 Prozent der Fahrzeuge, das am Straßenverkehr teilnimmt, ein Radio enthalten. [68] Somit bietet sich die Nutzung des RDS Dienstes TMC sehr gut an. Eine wirkliche Konkurrenz könnte lediglich ein weiterer, neuer Dienst des Radios sein. Dieser ist zurzeit nicht angedacht und wird es wohl in naher Zukunft auch nicht diskutiert werden. [67] Ein eher wahrscheinliches Szenario ist die Einstellung bzw. Ablöse von TMC durch einen neuen Dienst eines neuen Unterhaltungsmediums. Demnach ist anzunehmen, wenn eine alternative zum Radio Einzug in den Fahrzeugen hält, wird auch ein neuer Dienst entwickelt, der verkehrsbezogen Meldungen liefert. Diese Aussagen stützen sich auf die von James Burgess herausgegebenen Präsentationen, in seiner Tätigkeit als TMC Forums Koordinator. [67] [68] Sicher ist, dass eine wirkliche Alternative zu TMC zurzeit nicht existiert, auch wenn neue Geräte eine Alternative durch neue Dienste aus dem Netz bereitstellen könnten. So ist es bis zur Ablöse dieser Technik noch ein weiter Weg.

Angesichts des Ausbleibens einer baldigen Ablösung von TMC soll betrachtet werden, welche Perspektiven sich alternativ zu konventionellen TMC-Interpretation und die damit verbundene Verarbeitung ergeben.

Die Annahme, dass die TMC-Analyse auf deren Informationsgehalt selbst begrenzt zu sein scheint, ist falsch. Dies zeigt die Betrachtung der TMC-Meldungen untereinander. So lassen sich neue Informationen aus der Zusammenführung mehrerer Meldungen erstellen, wie es im Kapitel 6. Abschnitt 4. für Berlin gezeigt wird. Hier wird jede eingehende TMC-Meldung für den Bereich Berlin mit gleichen Ereignissen einer Gruppe zugeteilt. Aus der Anzahl aller Gruppen wird anschließend die Gruppe gewählt, die die meisten Meldungen enthält. Ableitend kann man aus dieser Gruppierung bestimmen, welche Ereignisse in dem beobachteten Bereich am häufigsten auftreten. Wenn man dieses Verfahren auf eine Straße anwendet, wie der A100 zum Beispiel, dann kann eine Aussage getroffen werden, mit welchen Ereignissen hier vermehrt zu rechnen ist. Ableitend für die Verkehrsanalyse ergeben sich ebenfalls neue Möglichkeiten. So ist es denkbar, dass eine Baustellendichte für einen gewissen Zeitraum aus den gruppierten TMC-Daten abgeleitet wird. Vergleicht man diese mit den Aufkommen von Verkehrsstörungen, könnten sich neue Erkenntnisse für die Stauvermeidung ableiten lassen. Diese Vermutung bezieht sich auf die Möglichkeit der Erstellung einer Erwartungshaltung, die direkt mit der Häufung spezieller Ereignisse verknüpft ist. Als Beispiel soll angenommen werden, dass die Verkehrsanalyse für die A100 auf Grund der neuen Datenbasis einen Zusammenhang zwischen dem Auftreten einer Verkehrsstörung und einer Baustellenhäufung pro 10km von mehr als 2 Stück entdeckt hat. Die entsprechende Konsequenz wäre eine Vermeidung dieser Häufung. Dieses fiktive Beispiel soll zeigen, wie eine neue Betrachtung und Zusammenführung von vorhandenen Informationen, neue Informationen generieren kann und eine Möglichkeit für zukünftige Betrachtungen zeigen.

13. Zusammenfassung

In diesem Kapitel wird in kurzen Schritten noch einmal die erbrachte Leistung rekapituliert und eine Kurzanleitung gegeben, um TMC-Meldungen zu empfangen, zu decodieren und entsprechend darzustellen. Dabei wird die folgende Erarbeitung allgemein gültig geschrieben, um das Verfahren auf eine andere Datenbasis anwendbar zu machen.

Datenempfang

Für die Anreicherung einer bestehenden Datenbasis mit TMC wird ein TMC-Empfänger benötigt. Dieser Empfänger sollte das GNS 3.0 Protokoll verwenden, um die GPSTMC-API von GNS zu nutzen. Grundsätzlich ist es egal wie die TMC-Meldungen empfangen werden, Wichtig ist, dass die TMC-Meldungen in lesbarer Form vorliegen und somit nach dem ISO Standards 14819-1 bis 14819-3 interpretiert werden können. [34] [35] [36] Bei der Automatisierung dieses Vorganges ist vor allem darauf zu achten, dass es eine lose Kopplung zwischen Empfangs-/ und Verarbeitungseinheit gibt. Die Schnittstelle selbst sollte dem ISO Standards für TMC entsprechen, wie es in dieser Arbeit gezeigt wird. Der Vorteil ist, dass das System beim Empfänger bzw. bei der Verarbeitungseinheit beliebig skalieren kann. Ferner ist es möglich Änderungen separat innerhalb der einzelnen Module vorzunehmen, ohne dass der Datenproduzent oder der Datenkonsument verändert werden müssen.

Decodierung

Bei der Decodierung der Daten müssen die Dateien „LCL10.1.D-110221.csv“ und „Event List_DE_4.01.xls“ in der neusten Ausgabe genutzt werden, um mit deren Hilfe den Location-Code (Ort) bzw. Event-Code (Ereignis) interpretieren zu können. Die Verwaltung und Herausgabe dieser beiden Dateien obliegt der Bundesanstalt für Straßenwesen. Auf der Seite der BASt ist es möglich, die aktuellen Decodierungsdateien zu erhalten. [32] Bei der Automatisierung der Decodierung sollte auf ein konstantes Zugriffsverhalten der geparsten Tabellen in der jeweiligen Programmiersprache geachtet werden, da für jede TMC-Meldung die entsprechenden Werte sonst erst aus der richtigen Datei gelesen und gefunden werden müssen. Dies beeinträchtigt den Verarbeitungsprozess. Sinnvoller ist ein einmaliges lesen der Dateien und der Ablage in einer geeigneten Datenstruktur, die ein gleiches Zugriffsverhalten sichern kann, wie eine „HashMap“ in Java. Ähnliche Strukturen sind ebenso in der Standard Bibliothek von C++ unter dem Namen „Map“ zu finden.

Anreicherung durch TMC

Bei der Erstellung einer neuen bzw. erweiterten Datenbasis sollte darauf geachtet werden, wo Gemeinsamkeiten der beiden Systeme bestehen. Bei TMC und FCD ist die Geschwindigkeit der gemeinsame Schnittpunkt, sofern diese als Zusatzinformation gesendet wird. Des Weiteren ist ebenso ein gemeinsamer Bezug aller anderen auftretenden Möglichkeiten des Informationsgehalts einer TMC-Meldung denkbar, wie Baustellen, Unfälle oder Großveranstaltungen. Wichtig ist, dass etwaige Gemeinsamkeiten genutzt werden können, um die Meldungen miteinander abzugleichen. Bei dem Abgleich selbst sollten vorhandene Strukturen sinnvoll genutzt werden, wie zum Beispiel Service oder Tabellen. Ferner sollte es vermieden werden, laufende Programme zu erweitern oder den bestehenden Prozessablauf der vorliegenden Verarbeitungskette zu unterbrechen, da dies zu unvorhergesehenen Fehlern führen kann. Außerdem sollte der Nutzen der Synergie im Fokus der Umsetzung stehen. So kann dank der Anreicherung von FCD durch TMC erstmals eine geschwindigkeitsbezogene Verkehrslage mit Ereignissen verknüpft werden, was zu einer Optimierung der Verkehrssituationsanalyse führt. Sicher ist bei jeder Anreicherung eines beliebigen Systems, dass die Wahrscheinlichkeit einer etwaigen Falschmeldung geringer wird, weil einer Verkehrssituation durch zwei unabhängige Systeme bestätigt wird. Bei widersprüchlichen Aussagen der vorliegenden Informationen, sollte das Alter der inkohärenten Daten verglichen werden. Dabei ist einer jüngeren TMC-Meldung mit einer unbestimmten Fehlerrate nur bedingt zu glauben. Wichtig ist, dass bei einer Gegenüberstellung der gemeinsamen Informationen die Fehlerrate mit berücksichtig werden sollte.

Darstellung der Datenbasis

Bei der Darstellung der zuvor erstellten Datenbasis sind viele Möglichkeiten in Betracht zu ziehen. Der Vorteil einer, wie bei dieser Arbeit genutzten servicegetriebenen Architektur ist es, dass beliebig viele Anwendungen den Dienst nutzen können. Der erstellte Remote Prozedur Call ermöglicht dem Nutzer den weiteren Betrieb, auch wenn der Dienst nicht funktionieren sollten, sofern dieser tatsächlich nur zur Darstellung genutzt wird. Somit ist es sinnvoll, bei der Darstellung der fusionierten Daten eine ähnliche Architektur anzustreben.

Diese 4 grundlegenden Schritte vom Empfang bis zur Darstellung einer durch TMC angereicherten Datenbasis, sollten unter Berücksichtigung der beschrieben Kriterien durchgeführt werden, um ein robustes, modulares und erweiterbares System zu schaffen.

Literaturverzeichnis

[1]. **Internationale Organisation für Normung -ISO.** its-actif.org. *its-actif.org.* [Online] [Zitat vom: 15. 06 2011.] http://www.its-actif.org/INTRANET/En/pages/ceb3d0fc3fc4114d.htm.

[2]. **NMEA.** kowoma.de. *kowoma.de.* [Online] [Zitat vom: 04. 07 2011.] http://www.kowoma.de/gps/zusatzerklaerungen/NMEA.htm.

[3]. **Austrosoft GmbH.** austrosoft.at. *austrosoft.at.* [Online] [Zitat vom: 27. 06 2011.] http://www.austrosoft.at/system_start.html.

[4]. **ESRI.** esri-germany.de. *esri-germany.de.* [Online] [Zitat vom: 28. 06 2011.] http://www.esri-germany.de/.

[5]. **Xu, Guochang.** *GPS -Theory, Algorithms and Applications.* s.l. : Springer, 2007. 978-3-540-72714-9.

[6]. **ISO/IEC 10918-1 JPEG.** w3.org. *w3.org.* [Online] [Zitat vom: 28. 06 2011.] http://www.w3.org/Graphics/JPEG/itu-t81.pdf.

[7]. **Wissen Media Verlag.** wissen.de. *wissen.de.* [Online] [Zitat vom: 17. 05 2011.] http://www.wissen.de/wde/generator/wissen/ressorts/bildung/index,page=1138096.html.

[8]. **Prof. Dr. -Ing. habil. Kurt Ackermann, Dipl. -Ing. Wilhelm Angenendt, Prof Dr. -Ing. Hartmut Beckedahl.** *Verkehr (Straßen, Schiene, Luft).* Kassel : Ernst & Sohn, 2001. 3-433-01576-7.

[9]. **http://www.navteq.com/.** navteq.com. *navteq.com.* [Online] [Zitat vom: 22. 08 2011.] http://www.navteq.com/.

[10]. **PostgreSql.** giswiki.org. *giswiki.org.* [Online] [Zitat vom: 22. 08 2011.] http://www.giswiki.org/wiki/PostGIS_Tutorial.

[11]. **UML Resource Page.** uml.org. *uml.org.* [Online] [Zitat vom: 13. 07 2011.] http://www.uml.org/.

[12]. **Bähr, Jürgen.** www.berlin-institut.org. *www.berlin-institut.org.* [Online] 2008. [Zitat vom: 17. 05 2011.] http://www.berlin-institut.org/fileadmin/user_upload/handbuch_texte/pdf_Baehr_Entwicklung_Urbanisierung.pdf.

[13]. **Pfister, U.** wiwi.uni-muenster.de. *wiwi.uni-muenster.de.* [Online] 2006. [Zitat vom: 17. 05 2011.] http://www.wiwi.uni-muenster.de/wisoge/md/studium/ws0506/gliederung.pdf.

[14]. **Schneider, Dr.-Ing. Dr.h.c Eckehard.** *Titel: Verkehrsleittechnik.* Ort: Berlin : Verlag: Springer, Jahr: 2007. ISBN: 9783540482963.

[15]. **LEIFI.** leifiphysik.de. *leifiphysik.de.* [Online] [Zitat vom: 17. 05 2011.] http://www.leifiphysik.de/web_ph11_g8/umwelt_technik/05schleifen/schleifen.htm.

[16]. **Hoppe-Johnen, Birgit.** duesseldorf.de. *duesseldorf.de.* [Online] 03 2003. [Zitat vom: 17. 05 2011.] http://www.duesseldorf.de/verkehrsmanagement/pdf/vid_det.pdf.

[17]. **Ruppe, Sten.** *Titel: DynVE-Verfahren und Vorrichtung zur Generierung von Verkehrsinformationen. Patentnummer: 102010018815,8* Deutsches Zentrum für Luft- und Raumfahrt e.V. Berlin, 2011.

[18]. **Sohr, Alexander.** elib.dlr.de. *elib.dlr.de.* [Online] [Zitat vom: 12. 06 2011.] http://elib.dlr.de/cgi/search/simple?screen=Public%3A%3AEPrintSearch&_action_search=Suchen&q_merge=ALL&q=dlr%20fcd&p_merge=ALL&p=&subjects_merge=ALL&date=&satisfyall=ALL&order=-date%2Fcreators_name%2Ftitle.

[19]. **GNS-Global Navigation Systems .** gns-gmbh.com. *gns-gmbh.com.* [Online] Global Navigation Systems . [Zitat vom: 21. 05 2011.] http://www.gns-gmbh.com/index.php?id=77.

[20]. **Böttcher, Christian Fochler & Frank.** www.tfh-wildau.de. *www.tfh-wildau.de.* [Online] 15. 11 2006. [Zitat vom: 24. 05 2011.] http://www.tfh-wildau.de/sbruntha/Material/ON/TM06/ON-RDS-TMS.pdf.

[21]. **Stul, Philip.** teslaplatten.ch. *teslaplatten.ch.* [Online] [Zitat vom: 24. 05 2011.] http://teslaplatten.ch/tp_biografie_radio_d.htm.

[22]. **Clemens Hahn Universität Ulm.** theorie.informatik.uni-ulm.de. *theorie.informatik.uni-ulm.de.* [Online] SS2006. [Zitat vom: 24. 05 2011.] http://theorie.informatik.uni-ulm.de/Lehre/SS6/seminar_info/hahn.pdf.

[23]. **Bernhard Weiskopf -Arbeitskreis der AGDX e. V.** ukwtv.de. *ukwtv.de.* [Online] [Zitat vom: 24. 05 2011.] http://www.ukwtv.de/artikel/technik/RDS-PI_in_D.pdf.

[24]. **T.R.N. Rao, E. Fujiwara.** *Titel: Error-Control Coding for Computer Systems.* s.l. : Verleger: Prentice Hall, Jahr: 1989. ISSN: 0018-9162.

[25]. **Lang, H.W.** *Titel: Algorithmen in Java. 2. Auflage.* Ort: Oldenbourg : s.n., Jahr: 2006. ISBN: 978-3-486-57938-3.

[26]. **2wcom.** 2wcom.com. *2wcom.com.* [Online] 2006. [Zitat vom: 25. 05 2011.] http://www.2wcom.com/fileadmin/redaktion/dokumente/Company/RDS_Basics.pdf.

[27]. **Rüegg, Dipl. Ing. David.** rds.cnlab.ch. *rds.cnlab.ch.* [Online] [Zitat vom: 25. 05 2011.] http://rds.cnlab.ch/RDStoWeb/.

[28]. **TMC-Liste.** navigatiehosting.nl. *navigatiehosting.nl.* [Online] [Zitat vom: 15. 06 2011.] http://www.navigatiehosting.nl/tmc/zenders/index/index.php.

[29]. **Roselius, Uwe.** surfmusik.de. *surfmusik.de.* [Online] [Zitat vom: 15. 06 2011.] http://www.surfmusik.de/.

[30]. **AviMedia - Akos Vida.** radioweb.de. *radioweb.de.* [Online] [Zitat vom: 15. 06 2011.] http://www.radioweb.de/stationen.html.

[31]. **TISA - Traveller Information Services Association.** lta.gov.sg. *lta.gov.sg.* [Online] [Zitat vom: 20. 06 2011.] http://www.lta.gov.sg/dbc/doc/TrafficMessageChannelBrochure.pdf.

[32]. **BASt - Bundesanstalt für Straßenwesen.** bast.de. *bast.de.* [Online] [Zitat vom: 20. 06 2011.] http://www.bast.de/cln_031/nn_42256/DE/Aufgaben/abteilung-f/referat-f4/Location-Code-List/location-code-list-start.html?__nnn=true.

[33]. **Google Maps LCL Beispiel.** maps.google.de. *maps.google.de.* [Online] [Zitat vom: 20. 06 2011.] http://maps.google.de/maps?hl=de&tab=wl.

[34]. **ISO 14819-1.** sirim.my. *sirim.my.* [Online] [Zitat vom: 02. 05 2011.] http://www.sirim.my/iscg/tc_g_3/14819-1.pdf.

[35]. **ISO 14819-2.** elearning.zhaw.ch. *elearning.zhaw.ch.* [Online] [Zitat vom: 02. 05 2011.] http://elearning.zhaw.ch/moodle/file.php/749/Spezifikationen/14819-2.pdf.

[36]. **ISO 14819-3.** sirim.my. *sirim.my.* [Online] [Zitat vom: 02. 05 2011.] http://www.sirim.my/iscg/tc_g_3/ISO14819_3.pdf.pdf.

[37]. **Hawking, Stephen.** *Das Universum in der Nussschale.* s.l. : Hoffmann u Campe Vlg GmbH, Oktober 2002. 3455093450.

[38]. **Sohr, Alexander.** *Floating Car Data.* [Präsentation] Berlin : DLR, 2011.

[39]. **Sönke Cordts, Gerold Blakowski und Gerhard Brosius.** *Datenbankschnittelle JDBC* . s.l. : Springer Verlag, 2011. 978-3-8348-8192-2_16.

[40]. **TU-Darmstadt Dr. Detlev Mares, Dr. Alexander von Lünen .** gis.geschichte.tu-darmstadt.de. *gis.geschichte.tu-darmstadt.de.* [Online] [Zitat vom: 28. 06 2011.] http://www.gis.geschichte.tu-darmstadt.de/index.php?id=1504.

[41]. **The Apache Software Foundation.** apache.org. *apache.org.* [Online] [Zitat vom: 28. 06 2011.] http://activemq.apache.org/.

[42]. **Redakteur xairro Vergleich von Programmiersprachen.** xairro.com. *xairro.com.* [Online] [Zitat vom: 04. 07 2011.] http://www.xairro.com/articles/programming/programmiersprachen-im-vergleich/.

[43]. **Navilock.** navilock.de. *navilock.de.* [Online] [Zitat vom: 04. 07 2011.] http://www.navilock.de/support/gruppen/15/Bluetooth_EmpfaengerSLASHreceiver/60318_BT-465UTE.html?show=datafile&type=3.

[44]. **The NMEA 0183 Protocol.** cs.put.poznan.pl. *cs.put.poznan.pl.* [Online] [Zitat vom: 04. 07 2011.] http://www.cs.put.poznan.pl/wswitala/download/pdf/NMEAdescription.pdf.

[45]. **GNS Global Navigation Systems.** *GNSTMC-API.* BERLN : s.n., 2010.

[46]. **JNI-API.** edm2.com. *edm2.com.* [Online] [Zitat vom: 12. 07 2011.] http://www.edm2.com/index.php/Java_JNI_API.

[47]. **JNA-API.** jna.java.net. *jna.java.net.* [Online] [Zitat vom: 17. 07 2011.] http://jna.java.net/javadoc/overview-summary.html.

[48]. **JNA Release Notes Bug Fixes.** jna.java.net. *jna.java.net.* [Online] [Zitat vom: 13. 07 2011.] http://jna.java.net/release-notes.html.

[49]. **Altova UML.** altova.com. *altova.com.* [Online] [Zitat vom: 20. 07 2011.] http://www.altova.com/de/umodel.html.

[50]. **Java HashMap API.** download.oracle.com. *download.oracle.com.* [Online] [Zitat vom: 22. 07 2011.] http://download.oracle.com/javase/1.4.2/docs/api/java/util/HashMap.html.

[51]. **Schmidt, Hans Albrecht.** *Objektorientierte Entwurfsmuster und Frameworks in der Informatik-Ausbildung an der Fachhochschule Konstanz* . s.l. : Springer Verlag, 1997. 002870050081.

[52]. **Gudenberg, Jochen Seemann und Jürgen Wolff von.** *Software-Entwurf mit UML 2 - Objektorientierte Modellierung mit Beispielen in Java.* s.l. : SpringerVerlag, 2006. 3-540-30950-0.

[53]. **IT Desy.** *Matlab Database Toolbox.* Berlin : s.n., 2009. 1214957862.

[54]. **Verkehrsmanagment Zentrale Berlin.** vmz-info.de. *vmz-info.de.* [Online] [Zitat vom: 16. 08 2011.] http://www.vmz-info.de/web/guest/about.

[55]. **DLR (interne Dokumente).** DLR Dokumente TS. Berlin : DLR, 2011.

[56]. **(W3C), World Wide Web Consortium.** www.w3.org. *www.w3.org.* [Online] [Zitat vom: 08. 09 2011.] http://www.w3.org/TR/ws-arch/.

[57]. —. w3.org. *w3.org.* [Online] [Zitat vom: 5. 10 2011.] http://www.w3.org/TR/wsdl.

[58]. **Narayanan, Srini.** *Simulation, verification and automated composition of web services.* New York : IEEE, 2002. 1-58113-449-5 .

[59]. **Curbera, F., et al.** *Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI.* s.l. : IEEE, 2002. 7222410 .

[60]. **Foster, H., et al.** *Model-based verification of Web service compositions* . London UK : IEEE, 2004. 0-7695-2035-9 .

[61]. **Fielding, Roy Thomas.** *Architectural Styles and the Design of Network-based Software Architectures.* UNIVERSITY OF CALIFORNIA, : IEEE, 2000. 1-524548-522547.

[62]. **RESTLet.** restlet.org. *restlet.org.* [Online] [Zitat vom: 12. 09 2011.] http://www.restlet.org/downloads/stable.

[63]. **Walther, Björn.** *Java 5: Taming the Tiger JSR-175.* Aargau : Fachhochschule Aargau (Schweiz), 2005.

[64]. **viralpatel.net.** viralpatel.net. *viralpatel.net.* [Online] [Zitat vom: 13. 09 2011.] http://viralpatel.net/blogs/2009/02/convert-jar-to-exe-executable-jar-file-to-exe-converting.html.

[65]. **Verkehrsinformation.** verkehrsinformation.de. *verkehrsinformation.de.* [Online] [Zitat vom: 10. 05 2011.] http://www.verkehrsinformation.de/.

[66]. **ADAC.** adac.de. *adac.de.* [Online] [Zitat vom: 10. 05 2011.] http://www.adac.de/reise_freizeit/verkehr/aktuelle_verkehrslage/default.aspx.

[67]. **Coordinator, James Burgess TMC Forum.** simbaproject.org. *simbaproject.org.* [Online] [Zitat vom: 5. 10 2011.] http://www.simbaproject.org/download/south_africa/SANE_July_07/1/8.pdf.

[68]. —. programtempo.si. *programtempo.si.* [Online] [Zitat vom: 6. 10 2011.] http://www.programtempo.si/media/1.20-.20tmc.20and.20tmc.20forum.20summary.20-.20for.20ljubliana.20february.202007.pdf.

[69]. **Runkler, Thomas A.** *Titel: Data Mining-Methoden und Algorithmen intelligenter Datenanalyse.* s.l. : Verlag: Vieweg+Teubner, Jahr: 2009. ISBN: 9783834808585.

[70]. **Stahel, Werner.** *Titel: Statistische Datenanalyse: Eine Einführung für Naturwissenschaftler.* s.l. : Verlag: Vieweg+Teubner, Jahr: 2007. ISBN: 383480410X.

[71]. **Kobbeloer, Detlef.** *Titel: Dezentrale Steuerung von Lichtsingnalangen in Urbanen Verkehrsnetzen.* Ort: Kassel : Verleger: Institut für Verkehrswesen Universität Kassel, Jahr: 2007. ISBN: 9783899583236.

[72]. **Google Tutorial.** google.com. *google.com.* [Online] [Zitat vom: 27. 06 2011.] http://code.google.com/intl/de-DE/apis/kml/documentation/kml_tut.html.

[73]. **Google Code Tutorial.** google.com. *google.com.* [Online] [Zitat vom: 28. 06 2011.] http://code.google.com/intl/de-DE/apis/kml/documentation/time.html.

[74]. **Winkler, Afschin Hormozdiary & Sebastian.** informatik.hu-berlin.de. *informatik.hu-berlin.de.* [Online] [Zitat vom: 05. 07 2011.] http://www2.informatik.hu-berlin.de/~swinkler/papers/GeschichtederVerschluesselung%28Hashfunktionen%29.pdf.

[75]. **Java Forum.** java-forum.org. *java-forum.org.* [Online] [Zitat vom: 13. 07 2011.] http://www.java-forum.org/allgemeine-java-themen/121532-jna-problem-kinect-mmd.html.

[76]. **Böhme, Dr. Andy.** *Fast FOURIER-Transformation.* 2005.

[77]. **Berlin, Taxi Zentrale.** *Antwort E-Mail zur Ausgabe des Taxi-Formulars.* Berlin : s.n., 2011.

[78]. **Anderegg, Frank.** *Raumbezogene Datentypen in SQL/MM Spatial und verwandten Standards.* [PDF] Rapperswil : Hochschule, 2010.

Anhang

Inhalt der CD:

- **Wichtige Dokumente der BASt:**
  - Event List_DE_4.01.xls
  - LCL10.1.D-110221.csv

- **Log-Dateien:**
  - TMC-Logdateien der Testphase vom 01.10.2011 bis 27.10.2011

- **Messkampagne:**
  - Rohdaten der Messkampagne der 5 Fahrzeuge

- **Programme:**
  - TMC Datenimport
  - TMC Prozessor
  - TMC Webservice

- **Quellen:**
  - PDF Quellen
  - Präsentationen

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Masterarbeit selbstständig angefertigt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Strausberg, 28.10.2011

.....

David Suske